HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Thomas Lange

Modellierung und Simulation des O_2- und CO_2-Stoffwechsels am isoliert perfundierten Herzen mit Anwendung der Bayes'schen Datenanalyse

2013

Fakultät **Mathematik/Naturwissenschaften/Informatik**

BACHELORARBEIT

Modellierung und Simulation des O_2- und CO_2-Stoffwechsels am isoliert perfundierten Herzen mit Anwendung der Bayes'schen Datenanalyse

Autor:
Thomas Lange

Studiengang:
Biotechnologie/Bioinformatik

Seminargruppe:
BI10w1-B

Erstprüfer:
Prof. Röbbe Wünschiers

Zweitprüfer:
Dr. Peter Dieterich

Mittweida, 2013

Bibliografische Angaben

Lange, Thomas: Modellierung und Simulation des O_2- und CO_2-Stoffwechsels am isoliert perfundierten Herzen mit Anwendung der Bayes'schen Datenanalyse, 77 Seiten, 12 Abbildungen, Hochschule Mittweida (FH), Fakultät Mathematik/Naturwissenschaften/Informatik

Bachelorarbeit, 2013

Englischer Titel

Modelling and Simulation of the O_2- and CO_2-exchange for the isolated perfused mouse heart under utilisation of the bayesian analysis

Kurzbeschreibung

In dieser Arbeit wird ein mathematisches Modell beschrieben, welches den O_2- und CO_2-Stoffwechsel am isolierten Herzen in den drei Kompartimenten Plasma, interstitiale Flüssigkeit und Parenchymalzellen beschreibt unter Berücksichtigung des Carbonat- und Phosphatpuffergleichgewichtes. Mit Hilfe dieses Modells wird nach möglichen Sensoren für eine experimentell festgestellte Flussänderung unter hyperkapnischen Bedingungen gesucht. Dazu werden die Zeitskalen der einzelnen Prozesse analysiert und es wird versucht einen Regelkreis zur Flusssteuerung in das Modell einzubringen. Zudem wird das Modell in MatLab implementiert, um dort mittels Bayes'scher Datenanalyse die Parameter des Modells an die experimentell gemessenen Werte anzupassen.

Danksagung

An dieser Stelle möchte ich mich bei Prof. Dr. Andreas Deußen und Dr. Peter Dieterich bedanken, für die Möglichkeit bedanken, meine Arbeit bei Ihnen am Physiologischen Institut der Medizinischen Fakultät Carl Gustav Carus der TU Dresden durchzuführen. Sie gaben mir, wie zudem auch Prof. Dr. Thomas Noll, gute Denkansätze, die maßgeblich zur Entstehung dieser Arbeit beigetragen haben. Ich möchte mich bei allen Mitarbeitern des Institutes, die ich während meiner Zeit dort kennen lernen durfte, bedanken für die angenehme und hilfsbereite Arbeitsatmosphäre. Zudem möchte ich mich bei Prof. Dr. Röbbe Wünschiers für seine weitreichende Unterstützung bezüglich der organisatorischen Bewältigung des Projektes bedanken. Zu guter Letzt möchte ich mich bei meinen Eltern dafür bedanken, dass Sie jederzeit für mich da waren, wenn ich Ihrer Hilfe bedurfte.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

1 Einleitung

Ziel dieser Arbeit ist es, das während der Praktikumsarbeit [25] erstellte Modell zur Simulation des Gasaustausches von O_2 und CO_2 unter Berücksichtigung von HCO_3^- und H^+ sowie eines Phosphatpuffers anzupassen, sodass es mit den am Institut für Physiologie der TU Dresden erhobenen experimentellen Daten, insbesondere denen von Heintz et al. [17] [18], verglichen werden kann. Diese Experimente beobachteten bei dem Wechsel der CO_2-Konzentration am Einfluss von physiologischen auf hyperkapnische Bedingungen eine biphasige Flussantwort (s. Abb. 1.1) [17] [18], wobei die genauen Mechanismen für die erste Phase unklar sind, während die zweite, langsamere Antwort auf NO-Freisetzung und der resultierenden Gefäßdilatation zurückgeführt werden konnten. Das entwickelte Modell soll hier helfen, Aufschluss über die Zeitskalen der ablaufenden Vorgänge zu erhalten und dadurch Rückschlüsse über mögliche Ursachen und mögliche Sensoren der Flussantwort zu finden. Für den primären Flussanstieg wird die Azidose als wichtie Ursache vermutet. Diverse Mechanismen molekularer Sensoren für CO_2, HCO_3^- und pH-Wert werden in der aktuellen Literatur noch ohne finales Ergebnis diskutiert [6] [35] [36].

Das verwendete Modell umfasst die Kompartimente Plasma (pl), interstiale Flüssigkeit (isf) und Parenchymalzellen (pc) in denen die Konzentrationen von O_2, CO_2, HCO_3^-, H^+, HPO_4^{2-} und $H_2PO_4^-$ als orts- und zeitabhängige Größen beschrieben werden.

Im Zuge dieser Arbeit werden die Zeitskalen des Modells bestimmt und empirisch eine Flussregulation gesucht, die das Modell mit den experimentellen Daten in Einklang bringt. Zudem wird ein einfacheres, ortsunabhängiges Modell hergeleitet, welches anschließend verwendet wird um die Parameter des Modells zur besseren Beschreibung der experimentellen Daten azupassen. Dies erfolgt mittels Bayes'scher Datenanalyse, welche in Form eines Metropolisalgorithmus in MatLab implementiert wurde. Zudem wurde die experimentelle Flussantwort formeltechnisch quantifiziert um darüber Aufschlüsse über die zeitlichen Zusammenhänge zu erhalten.

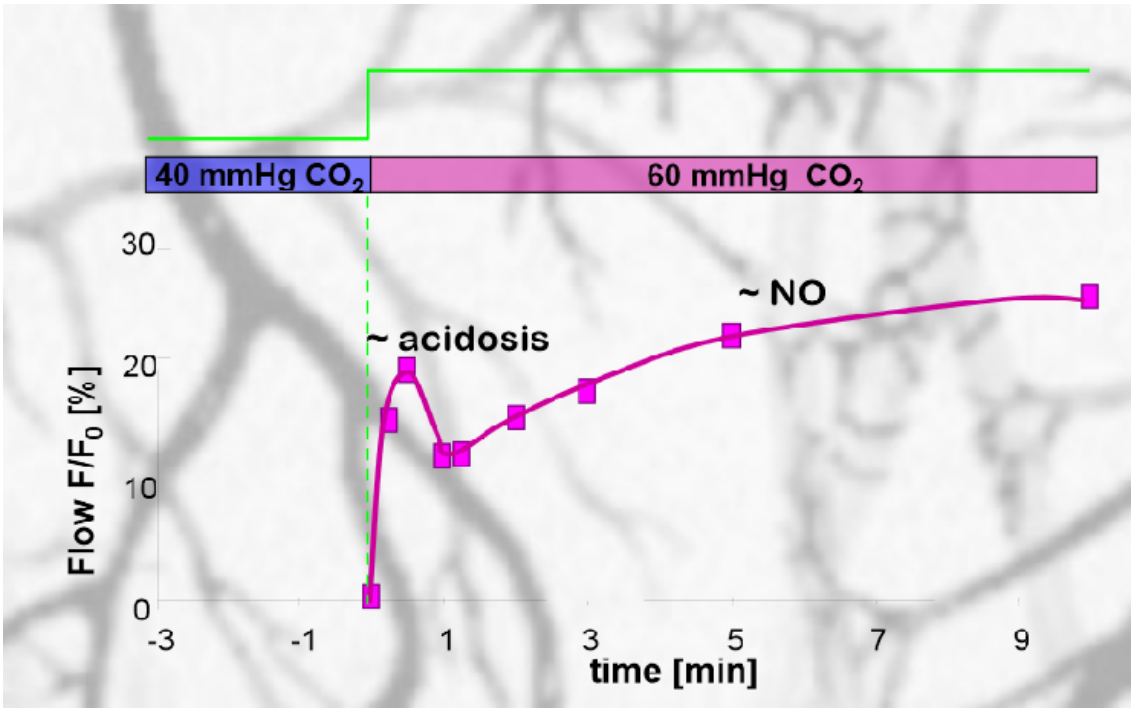

Abbildung 1.1: Verlauf der biphasigen Flussantwort, welche sich bei der Anpassung des einströmenden pCO_2 von 40 mmHg auf 60 mmHg am nach Langendorf isoliert perfundierten Herzen ergibt. [17]

2 Das mathematische Modell

Das Modell beschreibt die zeit- und ortsabhängige Konzentration von O_2, CO_2, HCO_3^-, H^+, $H_2PO_4^-$ und HPO_4^{2-} in den drei Kompartimenten Plasmaraum, interstitiale Flüssigkeit und Parenchymalzellen als gekoppelte partielle Differentialgleichungen. [7] Dabei werden folgende Prozesse berücksichtigt: In allen 3 Kompartimenten tritt eine Diffusion auf, welche im Plasma durch den konvektiven Pufferfluss ergänzt wird. Stoffaustausch zwischen den einzelnen Kompartimenten wird über Membranen mit einer konstanten Membranpermeabilität modelliert, sodass die Austauschrate linear abhängig ist von der Konzentrationsdifferenz der beiden Kompartimente und der Massenerhaltung folgt. Bei Ionen treten zusätzliche Effekte auf, welche bei den betrachteten Ionen über die Gleichgewichtskonstanten R_{cap} und R_{pc} in die Differentialgleichungen einfließen. Da physiologisch in den Parenchymalzellen eine bedeutend höhere Konzentration an Phosphationen vorkommt als im Interstitium und Plasma, jedoch im Modell keine Phosphatquelle berücksichtigt wird, wird dieser Effekt über die Startbedingungen des Modells realisiert, welche in den Parenchymalzellen eine Ausgangskonzentration von 60 mM vorgibt. [12] Um diesen Konzentrationsgradienten jedoch aufrecht erhalten zu können, wird kein Phosphataustausch zwischen Interstitium und Parenchymalzellen berücksichtigt. Physiologisch wird hingegen der Gradient hauptsächlich durch die Freisetzung von Phosphat beim ATP-Verbrauch gewährleistet. Der Zusammenhang der einzelnen Substanzen wird im Modell wie folgt angenommen: Sauerstoff wird in den Parenchymalzellen durch das Enzym Cytochrom-c-Oxidase umgesetzt. Dieser Vorgang wird über eine Michaelis-Menthen-Kinetik dargestellt. Im stöchiometrischen Verhältnis des respiratorischen Quotienten RQ wird eine entsprechende Menge an Kohlenstoffdioxid gebildet. Kohlenstoffdioxid steht durch die chemische Reaktion zu Kohlensäure in Gleichgewichtsrelation zu HCO_3^- und H^+ und somit zum pH-Wert. Dihydrogenphosphat und Hydrogenphosphat fungieren als pH-abhängiger Phosphatpuffer. Die Puffergleichgewichte werden dabei über das Massenwirkungsgesetz beschrieben. Zusätzlich zu diesen Übergängen wird Sauerstoff in den Parenchymalzellen zu einem gewissen Anteil an Myoglobin gebunden.

Für die Modellierung der einzelnen Prozesse fließen eine Vielzahl an Parametern in dieses System ein. Diese werden zunächst mit festen Werten [7] [25] gewählt, im Zuge dieser Arbeit jedoch teilweise mit Hilfe experimenteller Daten modifiziert.

Tabelle 2.1: Modellparameter des Basismodells

Symbol	Beschreibung	Wert	Einheit	Quelle
α_{O_2}	Löslichkeit von O_2 in Wasser bei Körpertemperatur	1,46*10^{-6}	M $mmHg^{-1}$	[16]

Tabelle 2.1 – *Fortsetzung*

Symbol	Beschreibung	Wert	Einheit	Quelle
α_{CO_2}	Löslichkeit von CO_2 in Wasser bei Körpertemperatur	3,27*10^{-5}	M $mmHg^{-1}$	[1]
β_{pl}	Pufferkapazität im Plasma	6*10^{-3}	M pH^{-1}	[21] [29]
β_{isf}	Pufferkapazität im isf	24*10^{-3}	M pH^{-1}	[21] [29]
β_{pc}	Pufferkapazität im pc	45*10^{-3}	M pH^{-1}	[21] [29]
C_{Mbpc}	Konzentration von Mb im Pc	0,5*10^{-3}	M	[8] [26]
${}^{s}D_r$	Diffusionskoeffizient von 's' in 'r'	1*10^{-4}	$cm^2\ s^{-1}$	[5] [8] [26]
F_{pl}	Blutfluss im Plasma	6	ml $min^{-1}\ g^{-1}$	[17]
kf_1	modifizierte Reaktionsrate der Vorwärtsreaktion der CO_2 Hydration	0,12	s^{-1}	[19] [20] [21]
kb_1	modifizierte Reaktionsrate der Rückwärtsreaktion der CO_2 Hydration	268000	$M^{-1}\ s^{-1}$	[19] [20] [21]
K_1	Gleichgewichtskonstante der CO_2 Hydration	4,467*10^{-7}	M	[19] [20] [21] [27]
K_m	Michaelis-Menden-Konstante der Cytochrome c-Oxidase	7*10^{-7}	M	[5] [14]
K_{pl}	Katalytischer Faktor der CO_2 Hydration im Plasma	100	dimensionslos	[19] [20] [21]
K_{isf}	Katalytischer Faktor der CO_2 Hydration im isf	5000	dimensionslos	[19] [20] [21]
K_{pc}	Katalytischer Faktor der CO_2 Hydration im pc	10000	dimensionslos	[19] [20] [21]
L	Kapillarlänge	1	mm	[2] [3] [8] [26]
P_{50}^{Mb}	p_{O_2}-Level bei der Mb zu 50% mit O_2 gesättigt ist	2,39	mmHg	[5] [32] [33]

Tabelle 2.1 – *Fortsetzung*

Symbol	Beschreibung	Wert	Einheit	Quelle
${}^{(O,C)}PS_{cap}$	Membranpermeabilität der Kapillarmembran für O_2 und CO_2	200	ml min^{-1} g^{-1}	[26]
${}^{(O,C)}PS_{pc}$	Membranpermeabilität der pc-Membran für O_2 und CO_2	2000	ml min^{-1} g^{-1}	[26]
${}^{(h,b)}PS_{cap}$	Membranpermeabilität der Kapillarmembran für HCO_3^- und H^+	20	ml min^{-1} g^{-1}	[26]
${}^{(h,b)}PS_{pc}$	Membranpermeabilität der pc-Membran für HCO_3^- und H^+	200	ml min^{-1} g^{-1}	[26]
RQ	Respiratorischer Quotient	0,8	dimensionslos	[22] [29]
R_{cap}	Gibbs-Donnan Verhältnis an der Kapillarmembran	0,63	dimensionslos	[21] [29] [37]
R_{pc}	Gibbs-Donnan Verhältnis an der pc-Membran	0,79	dimensionslos	[21] [29] [37]
v_{max}	Maximale Umsatzrate der Cytochrome c-Oxidase im pc	2,5*10^{-6}	mol min^{-1} g^{-1}	[5] [26]
V_{pl}	Plasmavolumen	0,07	ml g^{-1}	[26]
V_{isf}	Volumen des isf	0,2	ml g^{-1}	[26]
V_{pc}	Volumen des pc	0,7	ml g^{-1}	[26]
W_{pl}	Wasseranteil im Plasma	0,94	dimensionslos	[26]
W_{isf}	Wasseranteil im isf	0,92	dimensionslos	[26]
W_{pc}	Wasseranteil im pc	0,8	dimensionslos	[26]
VW_{pl}	wasseraktives Plasmavolumen	0,0658	ml g^{-1}	$V_{pl} * W_{pl}$
VW_{isf}	wasseraktives Volumen im isf	0,184	ml g^{-1}	$V_{isf} * W_{isf}$

Tabelle 2.1 – *Fortsetzung*

Symbol	Beschreibung	Wert	Einheit	Quelle
VW_{pc}	wasseraktives Volumen in den PC	0,56	ml g^{-1}	$V_{pc} * W_{pc}$
${}^{B}C_{pl_{in}}$	Konzentration von HPO_4^{2-} am Einfluss im Plasma	1*10^{-3}	M	[12]
${}^{B}C_{pc_0}$	Konzentration von HPO_4^{2-} in den pc	60*10^{-3}	M	[30]
${}^{(H,B)}D_r$	Diffusionsrate von $H_2PO_4^-$ und HPO_4^{2-}	1*10^{-4}	$cm^2\ sec^{-1}$	
kf_2	modifizierte Reaktionsrate der Vorwärtsreaktion des Phosphatpuffers	0,12	s^{-1}	
K_2	Gleichgewichtskonstante des Phosphatpuffers	6,2*10^{-8}	M	[27]
${}^{(H,B)}PS_{cap}$	Membranpermeabilität der Kapillarmembran für $H_2PO_4^-$ und HPO_4^{2-}	200	ml $min^{-1}\ g^{-1}$	

's' bezeichnet alle Substrate: O_2 (O), CO_2 (C), HCO_3^- (b), H^+ (h)

'r' bezeichnet alle Regionen: pl,isf und pc

Mit den so gewählten Parametern werden die im Modell berücksichtigten Prozesse über partielle gekoppelte Differentialgleichungen beschrieben. Es wird dazu jeweils beschrieben, wie sich die Konzentration ${}^{s}C_r$ des Substrates s (O=O_2, C=CO_2, b=HCO_3^-, B=HPO_4^{2-}, H=$H_2PO_4^-$ und h=H^+) in dem Kompartiment r (pl=Plasma, isf=Interstitium und pc= Parenchymzellen) im örtlichen und zeitlichen Verlauf durch die modellierten Prozesse ändert. Alle Konzentrationen ${}^{s}C_r$ sind von Ort x und der Zeit t abhängig. Die einzelnen Prozesse sind schematisch in Abb. 2.1 dargestellt. Dabei ergibt sich folgendes Differentialgleichungssystem:

$$\begin{aligned} \frac{\partial {}^{O}C_{pl}}{\partial t} &= -\frac{F_{pl}L}{V_{pl}}\frac{\partial {}^{O}C_{pl}}{\partial x} - \frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}({}^{O}C_{pl} - {}^{O}C_{isf}) + {}^{O}D_{pl}\frac{\partial^2 {}^{O}C_{pl}}{\partial x^2} \\ \frac{\partial {}^{O}C_{isf}}{\partial t} &= \frac{{}^{O}PS_{cap}}{V_{isf}W_{isf}}({}^{O}C_{pl} - {}^{O}C_{isf}) \end{aligned} \quad (2.1)$$

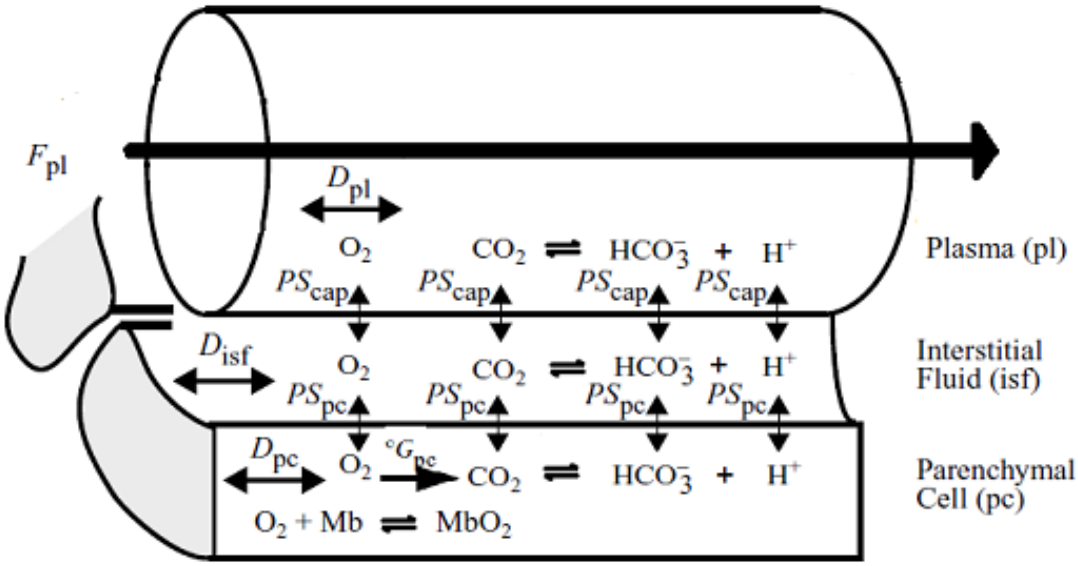

Abbildung 2.1: Symbolische Darstellung der Zusammenhänge im mathematischen Modells. Der Übersichtlichkeit halber ist die Reaktion $HPO_4^{2-} + H^+ \longleftrightarrow H_2PO_4$ sowie der Membrantransport der Hydrogenphosphationen vernachlässigt

$$-\frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}({}^{O}C_{isf} - {}^{O}C_{pc}) + {}^{O}D_{isf}\frac{\partial^2 {}^{O}C_{isf}}{\partial x^2} \tag{2.2}$$

$$\frac{\partial({}^{O}C_{pc} + C_{MbO_2})}{\partial t} = \frac{{}^{O}PS_{pc}}{V_{pc}W_{pc}}({}^{O}C_{isf} - {}^{O}C_{pc}) - \frac{V_{max}}{V_{pc} * W_{pc}} * \frac{{}^{O}C_{pc}}{{}^{O}C_{pc} + K_m} + {}^{O}D_{pc}\frac{\partial^2({}^{O}C_{pc} + C_{MbO_2})}{\partial x^2} \tag{2.3}$$

mit

$$C_{MbO_2} = C_{Mbpc} * \frac{K_{MbO_2} * {}^{O}C_{pc}}{1 + K_{MbO_2} * {}^{O}C_{pc}} \tag{2.4}$$

mit

$$K_{MbO_2} = \frac{1}{\alpha_{O_2} * P_{50}^{Mb}} \tag{2.5}$$

$$\frac{\partial {}^{C}C_{pl}}{\partial t} = -\frac{F_{pl}L}{V_{pl}}\frac{\partial {}^{C}C_{pl}}{\partial x} - \frac{{}^{C}PS_{cap}}{V_{pl}W_{pl}}({}^{C}C_{pl} - {}^{C}C_{isf}) + {}^{C}D_{pl}\frac{\partial^2 {}^{C}C_{pl}}{\partial x^2} + K_{pl}(kb_1^b C_{pl}^h C_{pl} - kf_1^C C_{pl}) \tag{2.6}$$

$$\frac{\partial {}^{C}C_{isf}}{\partial t} = \frac{{}^{C}PS_{cap}}{V_{isf}W_{isf}}({}^{C}C_{pl} - {}^{C}C_{isf}) - \frac{{}^{C}PS_{pc}}{V_{isf}W_{isf}}({}^{C}C_{isf} - {}^{C}C_{pc}) + {}^{C}D_{isf}\frac{\partial^2 {}^{C}C_{isf}}{\partial x^2} + K_{isf}(kb_1^b C_{isf}^h C_{isf} - kf_1^C C_{isf}) \tag{2.7}$$

$$\frac{\partial^C C_{pc}}{\partial t} = \frac{^C PS_{pc}}{V_{pc}W_{pc}}(^C C_{isf} - ^C C_{pc}) + RQ * \frac{V_{max}}{V_{pc} * W_{pc}} * \frac{^O C_{pc}}{^O C_{pc} + K_m}$$
$$+ ^C D_{pc}\frac{\partial^{2C} C_{pc}}{\partial x^2} + K_{pc}(kb_1^b C_{pc}^h C_{pc} - kf_1^C C_{pc}) \quad (2.8)$$

$$\frac{\partial^b C_{pl}}{\partial t} = -\frac{F_{pl}L}{V_{pl}}\frac{\partial^b C_{pl}}{\partial x} - \frac{^b PS_{cap}}{V_{pl}W_{pl}}(R_{cap}^b C_{pl} - ^b C_{isf}) + ^b D_{pl}\frac{\partial^{2b} C_{pl}}{\partial x^2}$$
$$-K_{pl}(kb_1^b C_{pl}^h C_{pl} - kf_1^C C_{pl}) \quad (2.9)$$

$$\frac{\partial^b C_{isf}}{\partial t} = \frac{^b PS_{cap}}{V_{isf}W_{isf}}(R_{cap}^b C_{pl} - ^b C_{isf}) - \frac{^b PS_{pc}}{V_{isf}W_{isf}}(R_{pc}^b C_{isf} - ^b C_{pc})$$
$$+ ^b D_{isf}\frac{\partial^{2b} C_{isf}}{\partial x^2} - K_{isf}(kb_1^b C_{isf}^h C_{isf} - kf_1^C C_{isf}) \quad (2.10)$$

$$\frac{\partial^b C_{pc}}{\partial t} = \frac{^b PS_{pc}}{V_{pc}W_{pc}}(R_{pc}^b C_{isf} - ^b C_{pc}) + ^b D_{pc}\frac{\partial^{2b} C_{pc}}{\partial x^2}$$
$$-K_{pc}(kb_1^b C_{pc}^h C_{pc} - kf_1^C C_{pc}) \quad (2.11)$$

$$\frac{\partial^B C_{pl}}{\partial t} = -\frac{F_{pl}L}{V_{pl}}\frac{\partial^B C_{pl}}{\partial x} - \frac{^B PS_{cap}}{V_{pl}W_{pl}}(R_{cap}^2 *^B C_{pl} - ^B C_{isf})$$
$$+ ^B D_{pl}\frac{\partial^{2B} C_{pl}}{\partial x^2} - (kb_2^B C_{pl}^h C_{pl} - kf_2^H C_{pl}) \quad (2.12)$$

$$\frac{\partial^B C_{isf}}{\partial t} = \frac{^B PS_{cap}}{V_{isf}W_{isf}}(R_{cap}^2 *^B C_{pl} - ^B C_{isf})$$
$$+ ^B D_{isf}\frac{\partial^{2B} C_{isf}}{\partial x^2} - (kb_2^B C_{isf}^h C_{isf} - kf_2^H C_{isf}) \quad (2.13)$$

$$\frac{\partial^B C_{pc}}{\partial t} = ^B D_{pc}\frac{\partial^{2B} C_{pc}}{\partial x^2} - (kb_2^B C_{pc}^h C_{pc} - kf_2^H C_{pc}) \quad (2.14)$$

$$\frac{\partial^H C_{pl}}{\partial t} = -\frac{F_{pl}L}{V_{pl}}\frac{\partial^H C_{pl}}{\partial x} - \frac{^H PS_{cap}}{V_{pl}W_{pl}}(R_{cap}^H C_{pl} - ^H C_{isf})$$
$$+ ^H D_{pl}\frac{\partial^{2H} C_{pl}}{\partial x^2} + (kb_2^B C_{pl}^h C_{pl} - kf_2^H C_{pl}) \quad (2.15)$$

$$\frac{\partial^H C_{isf}}{\partial t} = \frac{^H PS_{cap}}{V_{isf}W_{isf}}(R_{cap}^H C_{pl} - ^H C_{isf})$$
$$+ ^H D_{isf}\frac{\partial^{2H} C_{isf}}{\partial x^2} + (kb_2^B C_{isf}^h C_{isf} - kf_2^H C_{isf}) \quad (2.16)$$

$$\frac{\partial^H C_{pc}}{\partial t} = ^H D_{pc}\frac{\partial^{2H} C_{pc}}{\partial x^2} + (kb_2^B C_{pc}^h C_{pc} - kf_2^H C_{pc}) \quad (2.17)$$

$$\frac{\partial^h C_{pl}}{\partial t} = -\frac{F_{pl}L}{V_{pl}}\frac{\partial^h C_{pl}}{\partial x} - \frac{^h PS_{cap}}{V_{pl}W_{pl}}(^h C_{pl} - R_{cap}^h C_{isf}) + ^h D_{pl}\frac{\partial^{2h} C_{pl}}{\partial x^2}$$
$$-\frac{2,303}{\beta_{pl}}{}^h C_{pl}K_{pl}(kb_1^b C_{pl}^h C_{pl} - kf_1^C C_{pl})$$
$$-\frac{2,303}{\beta_{pl}}{}^h C_{pl}(kb_2^B C_{pl}^h C_{pl} - kf_2^H C_{pl}) \quad (2.18)$$

$$\begin{aligned}\frac{\partial^h C_{isf}}{\partial t} &= \frac{{}^hPS_{cap}}{V_{isf}W_{isf}}({}^hC_{pl} - R^h_{cap}C_{isf}) - \frac{{}^hPS_{pc}}{V_{isf}W_{isf}}({}^hC_{isf} - R^h_{pc}C_{pc}) \\ &\quad + {}^hD_{isf}\frac{\partial^{2h}C_{isf}}{\partial x^2} - \frac{2,303^{\,h}}{\beta_{isf}}\, C_{isf}K_{isf}(kb^b_1 C^h_{isf}C_{isf} - kf^C_1 C_{isf}) \\ &\quad - \frac{2,303^{\,h}}{\beta_{isf}}\, C_{isf}(kb^B_2 C^h_{isf}C_{isf} - kf^H_2 C_{isf}) \qquad (2.19)\end{aligned}$$

$$\begin{aligned}\frac{\partial^h C_{pc}}{\partial t} &= \frac{{}^hPS_{pc}}{V_{pc}W_{pc}}({}^hC_{isf} - R^h_{pc}C_{pc}) + {}^hD_{pc}\frac{\partial^{2h}C_{pc}}{\partial x^2} \\ &\quad - \frac{2,303^{\,h}}{\beta_{pc}}\, C_{pc}K_{pc}(kb^b_1 C^h_{pc}C_{pc} - kf^C_1 C_{pc}) \\ &\quad - \frac{2,303^{\,h}}{\beta_{pc}}\, C_{pc}(kb^B_2 C^h_{pc}C_{pc} - kf^H_2 C_{pc}) \qquad (2.20)\end{aligned}$$

Für die eindeutige Lösung der Differentialgleichungen werden zudem Randbedingungen benötigt. Dazu wird festgelegt, wie sich das System zum Zeitpunkt 0, sowie an den axialen Enden, das heißt dem arteriellen Einfluss und dem venösen Ausfluss, verhält. Prinzipiell ist die Wahl des Startzustandes - bis auf die Phosphatkonzentration in den Parenchymalzellen - beliebig, da sich nach einer gewissen Einstellzeit auf ein System einpegelt, dass ausschließlich durch die vorhandenen Prozesse und die darin einfließenden Parameter geprägt ist. Die benötigte Einstellzeit wird dabei durch eine Vorberechnung des Systems über mehrere Sekunden gewährleistet. Für eine möglichst schnelle und numerische stabile Einstellung, wird folgender Zustand für das Startsystem zum Zeitpunkt t_0 eingesetzt: Die Gase O_2 und CO_2 besitzen axial einen linearen Konzentrationsgradienten, der bei O_2 fällt und bei CO_2 mit dem Faktor RQ verechnet steigt. Die Konzentration der Ionen wird als axial konstant angenommen mit Ausgangskonzentrationen die sich aus den wirkenden Gleichgewichten ergeben.

$${}^sC_{pl}(t_0,0) = {}^sC_{in} \quad \text{für alle s} \qquad (2.21)$$

$${}^OC_{isf}(t_0,0) = {}^OC_{pl_0} - \frac{v_{max}}{{}^OPS_{cap}} \qquad (2.22)$$

$${}^OC_{pc}(t_0,0) = {}^OC_{pl_0} - \frac{v_{max}}{{}^OPS_{cap}} - \frac{v_{max}}{{}^OPS_{pc}} \qquad (2.23)$$

$${}^CC_{isf}(t_0,0) = {}^CC_{pl}(t_0,0) + \frac{RQ * v_{max}}{{}^CPS_{cap}} \qquad (2.24)$$

$${}^CC_{pc}(t_0,0) = {}^CC_{isf}(t_0,0) + \frac{RQ * v_{max}}{{}^CPS_{pc}} \qquad (2.25)$$

$${}^OC_r(t_0,L) = {}^OC_r(t_0,0) - \frac{v_{max}}{W_{pl} * F_{pl}} \quad \text{für alle r} \qquad (2.26)$$

$${}^CC_r(t_0,L) = {}^CC_r(t_0,0) + \frac{RQ * v_{max}}{W_{pl} * F_{pl}} \quad \text{für alle r} \qquad (2.27)$$

$$ {}^{(O,C)}C_r(t_0,x) = {}^{(O,C)}C_r(t_0,0) * \frac{L-x}{L} + {}^{(O,C)}C_r(t_0,L) * \frac{x}{L} \quad \text{für alle r,x} \tag{2.28} $$

$$ pH_{isf_0} = pH_{pl_0} + lg(R_{cap}) \tag{2.29} $$

$$ pH_{pc_0} = pH_{isf_0} + lg(R_{pc}) \tag{2.30} $$

$$ {}^{b}C_{r_0} = \frac{K_1 * {}^{C}C_{r_0}}{{}^{h}C_{r_0}} \quad \text{für alle r} \tag{2.31} $$

$$ {}^{B}C_{isf_0} = {}^{B}C_{pl_0} * R_{cap}^2 \tag{2.32} $$

$$ {}^{H}C_{r_0} = \frac{{}^{h}C_{r_0} * {}^{B}C_{r_0}}{KS_2} \quad \text{für alle r} \tag{2.33} $$

Für die axialen Randbedingungen werden sowohl am arteriellen Einfluss als auch venösen Ausfluss von-Neumann-Bedingungen angesetzt:

$$ {}^{s}D_{pl} * \frac{\partial {}^{s}C_{pl}(t,0)}{\partial x} = \frac{F_{pl} * L}{V_{pl}} * ({}^{s}C_{in}(t) - {}^{s}C_{pl}(t,0)) \quad \text{für alle s} \tag{2.34} $$

$$ \frac{\partial {}^{s}C_{isf,pc}(t,0)}{\partial x} = 0 \quad \text{für alle s} \tag{2.35} $$

$$ \frac{\partial {}^{s}C_{r}(t,L)}{\partial x} = 0 \quad \text{für alle s,r} \tag{2.36} $$

Das Modell wurde zunächst in jSim [39] implementiert und dort mittels des Toms-Solvers gelöst. Dieser Solver für steife, partielle Differentialgleichungen löst das System durch ein Gitterverfahren. Für die zeitliche Auflösung des Gitters wurde 1 Sekunde festgelegt, für die axiale Auflösung 0.02 mm. Der Solver berechnet dabei intern jedoch teilweise auf einer engeren Skala, damit das System stabil lösbar ist. t_0 wurde mit -90 Sekunden gewählt und die Simulationsergebnisse ab $t = 0$ wurden ausgewertet.

3 Zeitskalen des Systems

Für das Modell sollen die charakteristischen Zeitskalen bestimmt werden um darüber Rückschlüsse zu gewinnen, welche Prozesse aufgrund ihrer hohen Geschwindigkeit sich im quasi-steady-state befinden und welche Prozesse geeignet wären die Flussantwort, die im Zeitraum von bis zu 1 min verläuft [18], zeitlich passend zu regulieren.

3.1 Analytischer Ansatz

Für die charakteristischen Zeitskalen gibt es je nach Prozess verschiedene Definitionen und Berechnungsweisen [24]. Für die Berechnung der Zeitskalen werden zudem hier immer alle anderen Prozesse und deren Auswirkung auf das Einstellverhalten vernachlässigt.

3.1.1 Membranprozesse

Die Vorgänge an den Membranen lassen sich über die Relaxationszeit beschreiben. Diese gibt an, wie lange das System braucht, um bei einer Auslenkung aus dem Gleichgewicht diese Auslenkung um den Faktor e zu reduzieren. [24]

Zwischen den Membrankompartimenten stellt sich dabei ein Gleichgewicht ein, das wie folgt beschrieben werden kann: A $\underset{kf_-}{\overset{kf_+}{\rightleftharpoons}}$ B

In dem Modell werden diese Reaktionsgeschwindigkeiten als linear mit den Konzentrationen skalierend angenommen. Damit ergibt sich für die Differentialgleichung:

$$\frac{dA}{dt} = kf_- * B - kf_+ * A \tag{3.1}$$

Seien nun A_n und B_n die Gleichgewichtsstoffmengen des Substrates in den Kompartimenten A und B. δ(t) sei die Differenz zwischen aktueller Konzentration und Gleichgewichtskonzentration:

$$\delta(t) = A_n - A(t) \tag{3.2}$$

$$-\delta(t) = B_n - B(t) \tag{3.3}$$

Für den Gleichgewichtszustand ändern sich die Konzentrationen in A und B nicht mehr.

In der Differentialgleichung 3.1 ergibt dies:

$$\frac{dA_n}{dt} = \quad kf_- * B_n - kf_+ * A_n \quad = 0 \tag{3.4}$$

Es folgt:

$$\frac{dA}{dt} \overset{3.2}{=} \frac{d(A_n - \delta(t))}{dt} \tag{3.5}$$

$$\overset{3.1}{=} kf_- * B - kf_+ * A \tag{3.6}$$

$$\overset{3.2,3.3}{=} kf_- * (B_n + \delta(t)) - kf_+ * (A_n - \delta(t)) \tag{3.7}$$

$$\frac{d\delta(t)}{dt} \overset{3.4}{=} -\delta(t) * (kf_+ + kf_-) \tag{3.8}$$

$$\delta(t) = (A_n - A_0) * e^{-(kf_+ * kf_-)t} \tag{3.9}$$

Die Relaxationszeit τ resultiert somit als:

$$\tau = \frac{1}{kf_- + kf_+} \tag{3.10}$$

Diese Gleichung ist für die Gesamtheit der Membranprozesse gültig und deren Wert ist nur abhängig von den Parametern für k_+ und k_-. Für die im Modell verwendeten Membranen und Parameterwerte (s. Tabelle 2.1) ergeben sich dabei die in Tabelle 3.1 dargestellten Relaxationszeiten:

Tabelle 3.1: Zeitskalen der Membranprozesse

Prozess	kf_+	kf_-	τ **[ms]**
O_2-Transport zwischen pl und isf	$\frac{^{O}PS_{cap}}{VW_{pl}}$	$\frac{^{O}PS_{cap}}{VW_{isf}}$	14.54
O_2-Transport zwischen isf und pc	$\frac{^{O}PS_{pc}}{VW_{isf}}$	$\frac{^{O}PS_{pc}}{VW_{pc}}$	4.15
CO_2-Transport zwischen pl und isf	$\frac{^{C}PS_{cap}}{VW_{pl}}$	$\frac{^{C}PS_{cap}}{VW_{isf}}$	14.54
CO_2-Transport zwischen isf und pc	$\frac{^{C}PS_{pc}}{VW_{isf}}$	$\frac{^{C}PS_{pc}}{VW_{pc}}$	4.15
HCO_3^--Transport zwischen pl und isf	$\frac{R_{cap} \cdot {}^{b}PS_{cap}}{VW_{pl}}$	$\frac{^{b}PS_{cap}}{VW_{isf}}$	199.68

Tabelle 3.1 – *Fortsetzung*

Prozess	kf_+	kf_-	τ **[ms]**
HCO_3^--Transport zwischen isf und pc	$\frac{R_{pc} \cdot {}^bPS_{pc}}{VW_{isf}}$	$\frac{{}^bPS_{pc}}{VW_{pc}}$	49.16
H^+-Transport zwischen pl und isf	$\frac{{}^hPS_{cap}}{VW_{pl}}$	$\frac{R_{cap} \cdot {}^hPS_{cap}}{VW_{isf}}$	161.06
H^+-Transport zwischen isf und pc	$\frac{{}^hPS_{pc}}{VW_{isf}}$	$\frac{R_{pc} \cdot {}^hPS_{pc}}{VW_{pc}}$	43.78
$H_2PO_4^-$-Transport zwischen pl und isf	$\frac{R_{cap} \cdot {}^HPS_{cap}}{VW_{pl}}$	$\frac{{}^HPS_{cap}}{VW_{isf}}$	199.68
HPO_4^{2-}-Transport zwischen pl und isf	$\frac{R_{cap}^2 \cdot {}^BPS_{cap}}{VW_{pl}}$	$\frac{{}^BPS_{cap}}{VW_{isf}}$	261.21

Es ist festzustellen, dass die Zeitskalen aller Membranprozesse im Wesentlichen von dem Verhältnis der Membranpermeabilität zum wasseraktiven Volumen bestimmt sind. Bei den im Modell gewählten Parameterwerten liegen somit alle Zeitskalen im Bereich von 10 bis 200 ms, und sind damit viel zu schnell, um direkt ausschlaggebend für die Flussantwort zu sein.

3.1.2 Pufferprozesse

Für die Wirkung des Phosphat- und Bicarbonatpuffers wird die Antwortzeit [24] verwendet. Sie stellt eine Verallgemeinerung der Relaxationszeit auf mehr als das Ein-Edukt/Ein-Produkt-Gleichgewicht dar.

Die Puffergleichungen haben jeweils die Form $\mathrm{S}_1 \underset{kf_-}{\overset{kf_+}{\rightleftharpoons}} \mathrm{S}_2 + \mathrm{S}_3$

Durch das Massenwirkungsgesetz ergibt sich in der Differentialgleichung der folgende Zusammenhang (beim pH-Wert ergänzt durch eine zusätzliche unspezifische Pufferkapazität):

$$\frac{dS_1}{dt} = kf_- S_2 \cdot S_3 - kf_+ S_1 \tag{3.11}$$

$$\frac{dS_3}{dt} = \frac{2.303}{\beta_r} \cdot S_3 * (kf_+ S_1 - kf_1 S_2 \cdot S_3) \tag{3.12}$$

Wobei gilt:

$$kf_- = kb_2, \quad kf_+ = kp_2 \quad \text{für den Phosphatpuffer} \tag{3.13}$$

$$kf_- = K_r \cdot kb_1, \quad kf_+ = K_r \cdot kp_1 \quad \text{für den Carbonatpuffer} \tag{3.14}$$

In Verallgemeinerung der Zeitskalen für die Membranprozesse ist die Antwortzeit für Substrat j wie folgt definiert:

$$\frac{1}{\tau_j} = \sum_i n_{ij} \cdot \frac{\partial v_j}{\partial S_i} \tag{3.15}$$

n_{ij} bezeichnet dabei die Stöchiometriezahl in der chemischen Reaktion, v_j die Differentialgleichung, die die Änderung von Substrat j beschreibt.

Aus der Reaktionsgleichung und den Differentialgleichungen folgt sich folgender Zusammenhang für die Antwortzeit:

$$\frac{1}{\tau} = kf_-(S_2 + S_3) + kf_+ \quad \text{für C,b,H und B} \tag{3.16}$$

$$\frac{1}{\tau^h} = \frac{2.303}{\beta_r}(kf_+(S_1 - S_3) + kf_- S_3(S_3 + 2S_2)) \quad \text{für h} \tag{3.17}$$

Es ist festzustellen, dass hier die charakteristische Zeitskala abhängig von den Gleichgewichtskonzentrationen ist. Demzufolge wird die charakteristische Zeitskala sowohl für $pCO_{2_{in}}$ 40 mmHg als auch 60 mmHg durch die Simulation bestimmt.

Tabelle 3.2: Zeitskalen der Pufferprozesse

Prozess	τ_{40}	τ_{60}	τ^h_{40}	τ^h_{60}
Carbonat (pl)	2.49 μs	2.43 μs	51.56 ms	44.18 ms
Carbonat (isf)	78.6 ns	76.2 ns	4.09 ms	3.48 ms
Carbonat (pc)	49.5 ns	47.9 ns	3.83 ms	3.26 ms
Phosphat (pl)	333.6 μs	338.4 μs	6.99 s	6.45 s
Phosphat (isf)	840.1 μs	857.7 μs	44.25 s	40.72 s
Phosphat (pc)	11.4 μs	12.3 μs	0.88 s	0.84 s

Dabei fällt auf, dass die meisten Zeitskalen der Pufferreaktionen im Mikro- bis Nanosekundenbereich liegen. Demzufolge könnten diese Prozesse vermutlich auch als Quasi-steady-state aufgefasst werden. Die Zeitskala von H^+ in der Reaktion des Phosphatpuf-

fers beträgt mehrere Sekunden. Jedoch beträgt die Zeitskala von H^+ in der Reaktion des Carbonatspuffers nur wenige ms. Demzufolge ist anzunehmen, dass auftretende Änderungen hauptsächlich von dem schnelleren Carbonatpuffer gepuffert werden, als von dem im Vergleich dazu trägen Verhalten des Phosphatpuffers. Die Zeitskala von H^+ im Phosphatpuffer ist somit im Modell vernachlässigbar, da das System hauptsächlich durch den schnelleren Carbonatpuffer ausgeglichen wird, anstelle des Phosphatpuffers. Damit sind auch in den Pufferprozesse keine Vorgänge auffindbar, die die gesuchte Zeitskala direkt aufweisen.

3.1.3 Sonstige Prozesse

Für den Flussprozess kann die Transitzeit, das heißt die Zeit, die das Medium von dem arteriellen Zufluss zum venösen Abfluss benötigt, bestimmt werden. Die Formel hierfür lautet:

$$\tau_F = \frac{V_{pl}}{F} \tag{3.18}$$

Sie beträgt für die Modellparameter 0.7 s.

Für die Diffusion kann über die Bodenstein-Zahl Bo eine Zeitskala gefunden werden:

$$Bo = \frac{L * v}{D} = \frac{L * L * F}{V_{pl} * D} \tag{3.19}$$

Nimmt man die Bodensteinzahl nun als Maß der Geschwindigkeitsverhältnisse, so ergibt sich:

$$\frac{\tau_D}{\tau_F} = \frac{L^2 * F}{V_{pl} * D} \tag{3.20}$$

$$\tau_D = \frac{L^2}{D} \tag{3.21}$$

Diese Zeitskala der Diffusion τ_D, die als Zeit aufgefasst werden kann, bis ein Störsignal ausschließlich durch die Diffusion perfekt ausgeglichen wäre, beträgt 100 s.

3.1.4 Gemeinsame Zeitskalen

In allen bisherigen Betrachtungen wurden die einzelnen Prozesse jeweils unabhängig voneinander betrachtet. Durch die Störung eines Gleichgewichtes kommt es jedoch durch die Kopplungen auch zu einer Störung des Gleichgewichtes der anderen Teilprozesse. Durch diese Überlagerung von Störeffekten stellt sich das Gesamtsystem auf

einer anderen Zeitskala ein, als die Zeitskalen der einzelnen Prozesse vorgeben. Um diese Zeitskalen zu untersuchen wurden Simulationen mit sprunghaften Änderungen der Zuflusskonzentrationen der Substrate durchgeführt und untersucht, nach welcher zeitlichen Verzögerung sich das System zur Hälfte auf das neue Gleichgewicht eingestellt hat. Für die Simulationen wurde dabei die Konzentration jeweils von der Hälfte bzw. dem Doppelten auf die Konzentration unter den experimentellen Bedingungen verändert. Die zeitliche Auflösung wurde auf 0.2 s festgelegt. Der Zeitpunkt τ der halben Einstellung von der Ausgangskonzentration C_{eq_1} auf die neue Gleichgewichtskonzentration C_{eq_2} ergibt sich dabei über den folgenden Zusammenhang:

$$2 * (C(\tau) - C_{eq2}) \quad = \quad C_{eq1} - C_{eq2} \tag{3.22}$$

Für die Variation der Sauerstoffkonzentration ergibt sich, dass sich das Sauerstoffsystem nach 6 s zur Hälfte auf die neue Gleichgewichtskonzentration eingestellt hat. Für die anderen Konzentrationen konnte dabei jedoch keine Zeitskala bestimmt werden, da dort die Konzentrationsänderungen dabei so gering sind, dass diese in der Formel von numerischen Fehlern überlagert werden und ein diffuses Ergebnis entsteht.

Bei der Variation der anderen Konzentrationen ergaben sich die in Tabelle 3.3 aufgeführten Werte für die Zeitskalen der Halbeinstellung auf das neue Gleichgewicht:

Tabelle 3.3: Zeitskalen der Gesamtprozesse

varriierte Konz.	τ_{CO_2} **[s]**	τ_{H^+} **[s]**	$\tau_{HCO_3^-}$ **[s]**	$\tau_{H_2PO_4^-}$ **[s]**	$\tau_{HPO_4^{2-}}$ **[s]**
$CO_2\uparrow$	13	25	2.8	30	30
$CO_2\downarrow$	15.5	29	3	32.4	32.4
$H^+\uparrow$	34.8	30	33.6	36.2	35.8
$H^+\downarrow$	23.4	17.8	19.8	24.8	24.2
$HCO_3^-\uparrow$	3.8	13.8	2.4	19.8	19.4
$HCO_3^-\downarrow$	3	2-8*	2.8	17.4	16.8
$H_2PO_4^-\uparrow$	27.8	244	25.8	0.8	2.4
$H_2PO_4^-\downarrow$	31.4	29.2	29.8	0.8	2.4
$HPO_4^{2-}\uparrow$	30.2	27.6	28.4	2.4	0.6
$HPO_4^{2-}\downarrow$	29.8	27.2	28	2.4	0.6
Experimentbed.	16.4	29.4	2.6	33.2	32.6

*: Die Konzentration ist in diesem Intervall konstant C(τ)

Diese Zeitskalen der Gesamtprozesse liegen ca. im Intervall von 1 bis 30 Sekunden und könnten damit direkt mit der experimentellen Flussantwort korreliert sein. Somit ist die Flussantwort potenziell bei der Änderung der Konzentration zu suchen anstatt bei der Reaktionsdynamik eines einzelnen Teilprozesses. Bei der hyperkapnischen Reaktion auf Grund der Variation von CO_2 liegen dabei CO_2 und HCO_3^- an der experimentell bestimmten Zeitskala für das Mausherz von ca. 6-7 Sekunden [18] und sind somit Favoriten bei der Suche des Sensors für die auftretende Flussantwort.

3.2 Empirischer Ansatz

Die erste auftretende Flussantwort im Experiment ist ein temporärer Peak, der danach wieder auf sein Ausgangsniveau abfällt. Diese Flussantwort ist somit nicht durch die stationären Bedingungen des Systems gegeben, sondern basiert auf der Änderungsdynamik. Ein Feedback auf diese ist bisher im Modell nicht vorhanden. Infolgedessen wird versucht, die Flussantwort durch einen D-Regler darzustellen, dass heißt den auftretenden Fluss in Abhängigkeit zur differentiellen Änderung der Stoffkonzentration einer der Substrate zu setzen. Dabei wird die Stärke der Änderung der jeweiligen Sensorkonzentration mit dem Parameter $faktor$ skaliert.

$$F(t) \quad = \quad F_0 * (1 + \frac{\partial^s C_r}{\partial t} * faktor) \tag{3.23}$$

Das verwendete Programm jSim [39] lässt dabei an dieser Stelle keinen direkten Zugriff auf den Differentialterm zu, sodass anstelle dessen der Term durch den Differenzenquotient angenähert wird.

Alle Substratkonzentrationen in allen Kompartimenten wurden dabei als Kandidaten berücksichtigt und es wurde versucht, durch geeignete Verstärkung des Regelkreises ein Flusssignal zu erzeugen, dass sich in einem Peak widerspiegelt, dessen Maximum mit einem Wert von 116% des Ausgangsflusses nach 6-7 Sekunden eintritt und dessen Signal nach ca. 1 Minute wieder komplett abgeflacht ist, wie es in den Blockerexperimenten von Heintz et al. [18] am Mausherzen der Fall ist. Der Faktor wird jeweils so gewählt, dass die Peakhöhe bei ca. 116% liegt.

Dabei wurde festgestellt, dass Hydrogenphosphat und Dihydrogenphosphat aufgrund ihrer Trägheit und marginalen Änderung für die Flussantwort kaum in Frage kommen, die Konzentration von CO_2 und HCO_3^- in Plasma und Interstitium für die Antwort geeignet sein könnten. Jedoch ist die Eignung dabei stark von der konkreten Wahl der Systemparameter sowie der Schärfe des eingehenden Einflusssignals abhängig. Durch die Sprungfunktion ist der Maximalpeak zudem deutlich zu spitz zulaufend.

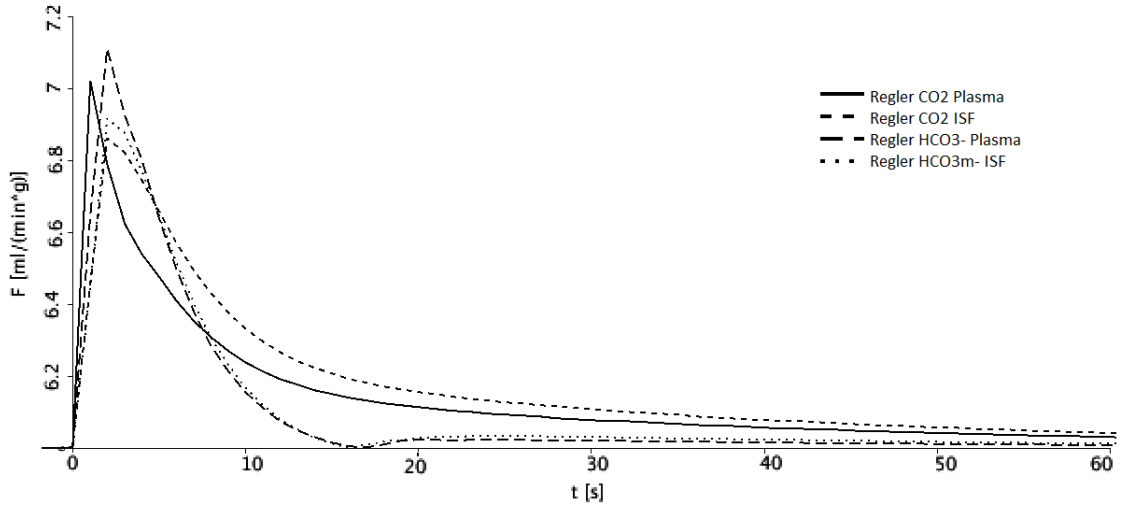

Abbildung 3.1: Eintretende primäre Flussantwort des differentiellen Regelkreises in Abhängigkeit des gewählten Sensors

Für die Schärfe des Einflusssignals wird dazu das Einflussrohr als zusätzlicher diffusiver Raum berücksichtigt, wie es von King et al. [23] beschrieben wird. Dabei wird das tatsächliche Einflusssignal aus dem idealen Einflusssignal durch ein zusätzliches Delay für die Durchströmungszeit den folgenden funktionalen Zusammenhang, ein diffusiver Operator 4. Ordnung, beschrieben:

$$\frac{d^2C_2}{dt^2} + 2 * \chi_1 * \omega_1 * \frac{dC_2}{dt} + \omega_1^2 * C_2 = \omega_1^2 * C_1 \tag{3.24}$$

$$\frac{d^2C_{in}}{dt^2} + 2 * \chi_2 * \omega_2 * \frac{dC_{in}}{dt} + \omega_2^2 * C_{in} = \omega_2^2 * C_2 \tag{3.25}$$

Wobei C_1 die zeitlich verschobene Sprungfunktion darstellt und C_{in} die tatsächliche einfließende Konzentration beschreibt. C_2 stellt ein vorläufiges, diffuses Signal dar, das lediglich für die effektive Berechnung von C_{in} benötigt wird (s. Abbildung 3.2).

Die Parameter ergeben sich wie folgt:

$$t_{mean} = \frac{V_{ein}}{F} \tag{3.26}$$

$$t_p = \frac{D_{ein}}{0.48} * t_{mean} \tag{3.27}$$

$$\omega_1 = \frac{2 * \pi}{t_p} \tag{3.28}$$

$$\omega_2 = 1.82 * \omega_1 \tag{3.29}$$

$$\chi_1 = 0.95 \tag{3.30}$$

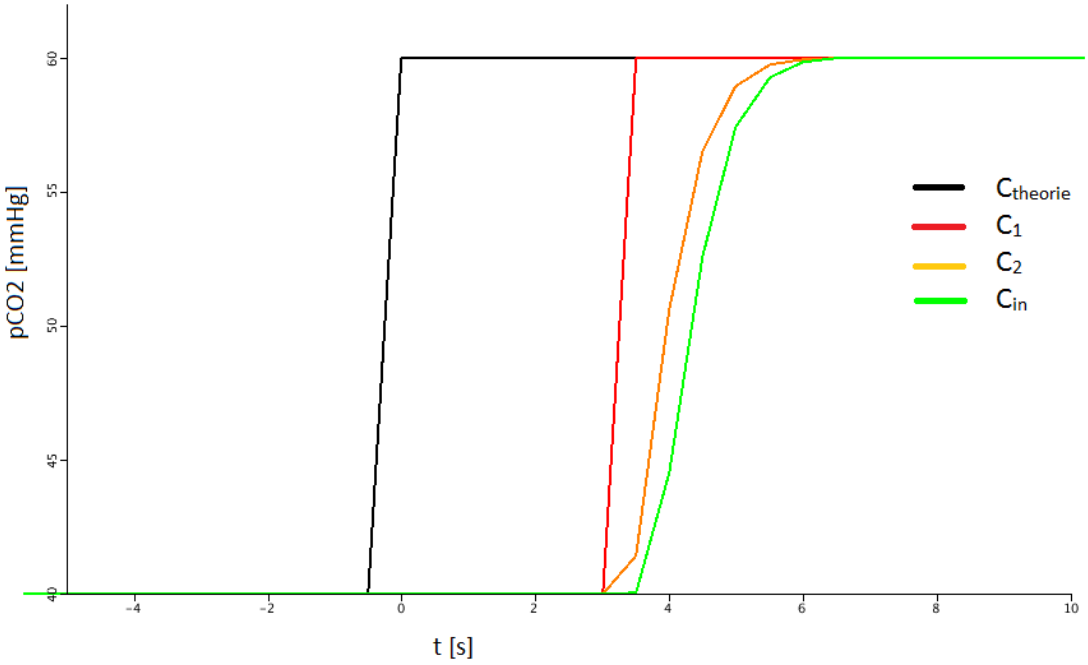

Abbildung 3.2: Auswirkung des diffusiven Einflussventils nach King [23]. Es sind die ursprüngliche theoretische Sprungfunktion (schwarz), die zeitlich verschobene Sprungfunktion (rot) sowie die diffusive Einflussfunktion (grün) dargestellt, sowie die für die Berechnung benötigte Zwischenfunktion C_2

$$\chi_2 = 0.8 \tag{3.31}$$

Das Volumen des Einflussventils V_{ein} beträgt ca. 0.5 ml, für die Diffusion wird ein Wert von 0.35 festgelegt.

Es ist festzustellen, dass der Initialpeak deutlich sanfter verläuft, sodass die erhaltenen Simulationsergebnisse besser mit den experimentellen Daten übereinstimmen. Jedoch tritt durch den zusätzlichen Durchströmungsraum eine zeitliche Verschiebung statt, die in der Form experimentell nicht beobachtbar war. Auch stimmt der prinzipielle Kurvenverlauf bezüglich der Breite und dem Abflachverhalten nicht mit der experimentellen Funktion überein. Demzufolge ist die Annahme, dass der Einfluss diffuser als eine perfekte Sprungfunktion ist, sinnvoll. Die Ursache für die Diffusivität ist jedoch vermutlich eher im menschlichen Umschalten des Ventils und durch Mischeffekte beider Einflüsse während des Umschaltens zu sehen als im Volumen des Einflussventils. Für eine exaktere Aussage bezüglich der Diffusivität am Einfluss wären Experimente mit kontinuierlicher CO_2-Messung am Einfluss nötig.

Bei der Unterscheidung zwischen CO_2 und HCO_3^- bezüglich der Eignung als Regelkreis bietet die Diffusivität des Einflusses leider keinen Mehrwert.

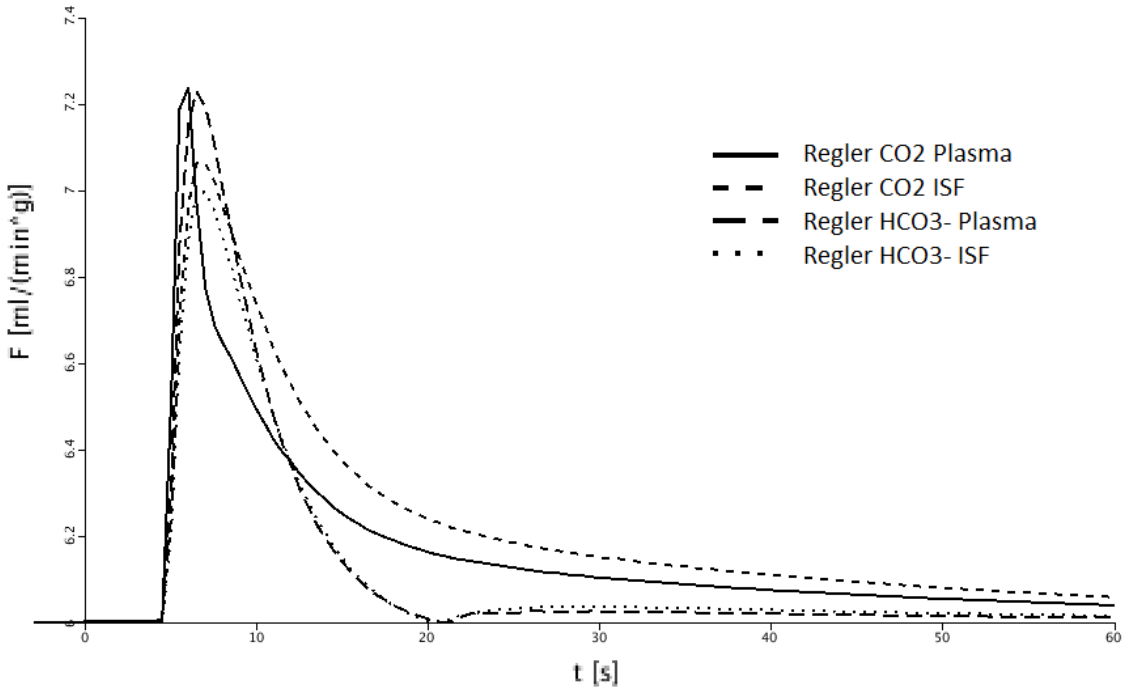

Abbildung 3.3: Eintretende primäre Flussantwort des differentiellen Regelkreises in Abhängigkeit des gewählten Sensors unter Berücksichtigung von diffusiven Effekten im Einflussventil

4 Grundlagen der Parametervorhersage

Es ist festzustellen, dass für exaktere Vorhersagen über den zeitlichen Verlauf des Modells die einzelnen systemrelevanten Parameter durch experimentelle Daten anzupassen sind, anstatt sie aus anderen Modellen und Literaturstellen zu übernehmen.

Um für die Parameteroptimierung Aussagen über die Wahrscheinlichkeitsverteilung/Streuung der Parameter anhand der Daten gewährleisten zu können, wurde für die Parametervorhersage eine Bayes'sche Datenanalyse mittels Metropolisalgorithmus implementiert. Da sich die Einbindung von Skripten zum Aufruf der Simulation mit verschiedenen Parametern als schwierig gestaltete, wurde das Modell für eine Bayes'sche Datenanalyse in eine andere Programmiersprache überführt.

4.1 Modellüberführung von jSim in Python und MatLab

Es wurde in der Programmiersprache Python mit dem Package Fipy [13] zum Lösen von Differentialgleichungen implementiert. Dabei ist jedoch festzustellen gewesen, dass sich das System aufgrund der stark unterschiedlichen Zeitskalen instabil verhält und das zeitliche Raster deswegen enger gelegt werden muss. Aufgrund der sowohl zeitlichen als auch axialen Verteilung des Systems, stellte sich dabei ein Gitter mit 101 axialen Datenpunkten und einer zeitlichen Auflösung von 0.1 ms als numerisch stabil heraus. Durch diesen Übergang auf ein sehr kleines Zeitgitter gegenüber der Analyse mit jSim kommt es dabei zu einem Ansteig der Laufzeit, sodass die Simulation eines einzelnen Parametersets mit der vorhandenen Rechentechnik in ca. 24 Stunden realisiert werden könnte. Da für den Metropolisalgorithmus jedoch für jeden Parameter mehrere tausend Werte verwendet werden, war es erforderlich, das Modell bezüglich seiner Laufzeit zu optimieren.

Dazu wurde durch Integration über den Ort ein ortsunabhängiges Stirred-Tank-Modell aus dem Krogh-Zylindermodell hergeleitet. Zur weiteren Vereinfachung wurden die Differentialgleichungen anschließend als Differenzenquotienten betrachtet und das so erhaltene Modell wurde in Matlab [38] implementiert. Durch die effizienten Vektoroperationen in MatLab wurde das Modell derartig implementiert, dass mehrere Parametersets in Vektorform parallel prozessiert werden können. Dadurch reduzierte sich die Rechenzeit auf 30 Sekunden für die parallele Simulation von 100 Parametersets.

Dabei war als erstes die Frage zu klären, welche Abhängigkeiten zwischen den Parametern herrschen. Dabei wurde festgestellt, dass die Volumina $V_{isf,pc}$ und Wasseranteile $W_{isf,pc}$ zueinander abhängig sind und sie wurden durch einen neuen Effektivparameter $VW_{isf,pc}$ ersetzt. Ebenso besteht eine Abhängigkeit zwischen der Reaktionsgeschwin-

digkeit des Carbonatpuffers und den katalytischen Faktoren der Carboanhydrase. Dabei wurden die Produkte aus katalytischem Faktor und Ausgangsreaktionsgeschwindigkeit in eine Effektivgeschwindigkeit der Carbonatpuffers überführt. Damit wurde ein System geschaffen, dass nun noch 32 anstatt vorher 35 Parametern enthält.

4.2 Überführung von Krogh-Zylinder-Modell auf Stirred Tank Modell

Folgende neue Funktionen für die mittlere Konzentrationen des Kompartiments werden für ein effektives Stirred Tank Modell eingeführt:

$$ {}^{s}\bar{C}_r(t) = \frac{1}{L}\int_0^L {}^{s}C_r(t,x)dx \quad \text{für alle s,r} \tag{4.1} $$

$$ \bar{C}_{MbO_2}(t) = \frac{1}{L}\int_0^L C_{MbO_2}(t,x)dx \tag{4.2} $$

Durch Integration über x und Division durch L ergibt sich aus dem Sauerstoffdifferentialgleichungssystem für das Krogh-Zylinder-Modell folgendes neue Differentialgleichungssystem für die Sauerstoffkonzentrationen:

$$ \begin{aligned} \frac{\partial}{\partial t}(1/L * \int_0^L {}^{O}C_{pl}dx) &= -\frac{F_{pl}}{V_{pl}}\int_0^L \frac{\partial {}^{O}C_{pl}}{\partial x}dx \\ &\quad -\frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}(\frac{1}{L}\int_0^L {}^{O}C_{pl}dx - \frac{1}{L}\int_0^L {}^{O}C_{isf}dx) \\ &\quad +{}^{O}D_{pl}\frac{1}{L}\int_0^L \frac{\partial^2 {}^{O}C_{pl}}{\partial x^2}dx \end{aligned} \tag{4.3} $$

$$ \begin{aligned} \frac{\partial}{\partial t}(1/L * \int_0^L {}^{O}C_{isf}dx) &= \frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}(\frac{1}{L}\int_0^L {}^{O}C_{pl}dx - \frac{1}{L}\int_0^L {}^{O}C_{isf}dx) \\ &\quad -\frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}(\frac{1}{L}\int_0^L {}^{O}C_{isf}dx - \frac{1}{L}\int_0^L {}^{O}C_{pc}dx) \\ &\quad +{}^{O}D_{isf} * \frac{1}{L} * \int_0^L \frac{\partial^2 {}^{O}C_{isf}}{\partial x^2}dx \end{aligned} \tag{4.4} $$

$$\frac{\partial}{\partial t}(1/L * \int_0^L ({}^{O}C_{pc} + C_{MbO_2})dx) = \frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}(\frac{1}{L}\int_0^L {}^{O}C_{isf}dx - \frac{1}{L}\int_0^L {}^{O}C_{pc}dx) - \frac{v_{max}}{V_{pc} * W_{pc} * L} * \int_0^L \frac{{}^{O}C_{pc}}{{}^{O}C_{pc} + K_m}dx + {}^{O}D_{pc} * \frac{1}{L}\int_0^L \frac{\partial^2({}^{O}C_{pc} + C_{MbO_2})}{\partial x^2}dx \quad (4.5)$$

mit

$$C_{MbO_2} = C_{Mbpc} * \frac{K_{MbO_2} * {}^{O}C_{pc}}{1 + K_{MbO_2} * {}^{O}C_{pc}} \quad (4.6)$$

mit

$$K_{MbO_2} = \frac{1}{\alpha_{O_2} * P_{50}^{Mb}} \quad (4.7)$$

Durch Einsetzen der Substitutionen aus 4.1 und Integration geht das das System von partiellen Differentialgleichungen in gewöhnliche Differentialgleichungen über und es ergibt sich:

$$\frac{d}{dt}{}^{O}\bar{C}_{pl} = -\frac{F_{pl}}{V_{pl}}({}^{O}C_{pl_L} - {}^{O}C_{pl_0}) - \frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}({}^{O}\bar{C}_{pl} - {}^{O}\bar{C}_{isf}) + \frac{{}^{O}D_{pl}}{L}(\frac{\partial {}^{O}C_{pl_L}}{\partial x} - \frac{\partial {}^{O}C_{pl_0}}{\partial x}) \quad (4.8)$$

$$\frac{d}{dt}{}^{O}\bar{C}_{isf} = \frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}({}^{O}\bar{C}_{pl} - {}^{O}\bar{C}_{isf}) - \frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{isf} - {}^{O}\bar{C}_{pc}) + \frac{{}^{O}D_{isf}}{L} * (\frac{\partial {}^{O}C_{isf_L}}{\partial x} - \frac{\partial {}^{O}C_{isf_0}}{\partial x}) \quad (4.9)$$

$$\frac{d}{dt}({}^{O}\bar{C}_{pc} + \bar{C}_{MbO_2}) = \frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{isf} - {}^{O}\bar{C}_{pc}) - \frac{v_{max}}{V_{pc} * W_{pc} * L} * \int_0^L \frac{{}^{O}C_{pc}}{{}^{O}C_{pc} + K_m}dx + \frac{{}^{O}D_{pc}}{L}(\frac{\partial({}^{O}C_{pc_L} + C_{MbO_{2_L}})}{\partial x} - \frac{\partial({}^{O}C_{pc_0} + C_{MbO_{2_0}})}{\partial x}) \quad (4.10)$$

Durch Einsetzen der Randbedingungen des Krogh-Zylinder-Modells ergibt sich daraus:

$$\frac{d}{dt}{}^{O}\bar{C}_{pl} = -\frac{F_{pl}}{V_{pl}}({}^{O}C_{pl_L} - {}^{O}C_{pl_0}) - \frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}({}^{O}\bar{C}_{pl} - {}^{O}\bar{C}_{isf})$$

$$+(0-\frac{F_{pl}}{V_{pl}}*({}^{O}C_{pl_0}-{}^{O}C_{pl_{in}}) \tag{4.11}$$

$$\frac{d}{dt}{}^{O}\bar{C}_{isf} = \frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}({}^{O}\bar{C}_{pl}-{}^{O}\bar{C}_{isf})-\frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{isf}-{}^{O}\bar{C}_{pc}) + \frac{{}^{O}D_{isf}}{L}*(0-0) \tag{4.12}$$

$$\frac{d}{dt}({}^{O}\bar{C}_{pc}+\bar{C}_{MbO_2}) = \frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{isf}-{}^{O}\bar{C}_{pc}) - \frac{v_{max}}{V_{pc}*W_{pc}*L}*\int_0^L \frac{{}^{O}C_{pc}}{{}^{O}C_{pc}+K_m}dx + \frac{{}^{O}D_{pc}}{L}(0-0) \tag{4.13}$$

Beziehungsweise vereinfacht:

$$\frac{d}{dt}{}^{O}\bar{C}_{pl} = \frac{F_{pl}}{V_{pl}}({}^{O}C_{pl_{in}}-{}^{O}C_{pl_L})-\frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}({}^{O}\bar{C}_{pl}-{}^{O}\bar{C}_{isf}) \tag{4.14}$$

$$\frac{d}{dt}{}^{O}\bar{C}_{isf} = \frac{{}^{O}PS_{cap}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{pl}-{}^{O}\bar{C}_{isf})-\frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{isf}-{}^{O}\bar{C}_{pc}) \tag{4.15}$$

$$\frac{d}{dt}({}^{O}\bar{C}_{pc}+\bar{C}_{MbO_2}) = \frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{isf}-{}^{O}\bar{C}_{pc}) - \frac{v_{max}}{V_{pc}*W_{pc}*L}*\int_0^L \frac{{}^{O}C_{pc}}{{}^{O}C_{pc}+K_m}dx \tag{4.16}$$

Es ist festzustellen, dass sich der sogenannte Gulosity-Term für den Sauerstoffverbrauch dabei nicht exakt überführen lässt, sodass eine gute Näherung für diesen hergeleitet wurde. Dafür wird folgende Substitution eingeführt, welche die ortsabhängige Konzentration in die mittlere Konzentration und eine ortsabhängige Fluktuation zerlegt:

$${}^{O}C_{pc}(t,x) = {}^{O}\bar{C}_{pc}(t)+\delta{}^{O}C_{pc}(t,x) \tag{4.17}$$

Dabei gilt:

$$\begin{aligned}\frac{1}{L}*\int_0^L \delta{}^{O}C_{pc}dx &= \frac{1}{L}*\int_0^L {}^{O}C_{pc}dx-\frac{1}{L}*\int_0^L {}^{O}\bar{C}_{pc}dx \\ &= {}^{O}\bar{C}_{pc}-{}^{O}\bar{C}_{pc} \\ &= 0\end{aligned} \tag{4.18}$$

Aus der Taylerentwicklung von C(t,x) in der Umgebung von ${}^{O}\bar{C}_{pc}$ folgt:

$$\begin{aligned}
\frac{{}^{O}C_{pc}}{{}^{O}C_{pc}+K_m} &= 1-\frac{K_m}{{}^{O}C_{pc}+K_m} \\
&= 1-\frac{K_m}{{}^{O}\bar{C}_{pc}+\delta^{O}C_{pc}+K_m} \\
&= 1-\frac{K_m}{{}^{O}\bar{C}_{pc}+K_m} \\
&+ \frac{K_m}{({}^{O}\bar{C}_{pc}+K_m)^2}*\delta^{O}C_{pc} \\
&+ \sum_{i=2}^{\infty}(-1)^{(i+1)}*\frac{K_m}{({}^{O}\bar{C}_{pc}+K_m)^{i+1}}*(\delta^{O}C_{pc})^i
\end{aligned} \tag{4.19}$$

Diese unendliche Reihe konvergiert, da aufgrund der physiologischen Randbedingungen die Fluktuation betragsmäßig stets geringer als die mittlere Konzentration sein sollte.

Unter Anwendung der aus den Berechnungen des stationären Falles ermittelten Lösung, welche in der Praktikumsarbeit [25] ermittelt wurde, wird folgende Näherung angewendet:

$${}^{O}C_{pc} \approx a*x+b \tag{4.20}$$

mit

$$a = -\frac{v_{max}}{F_{pl}*L*W_{pl}} \tag{4.21}$$

$$b = {}^{O}C_{pl_{in}}-\frac{v_{max}}{{}^{O}PS_{cap}}-\frac{v_{max}}{{}^{O}PS_{pc}} \tag{4.22}$$

Und somit:

$${}^{O}\bar{C}_{pc} = \frac{1}{L}*\int_0^L {}^{O}C_{pc}dx \approx a*L/2+b \tag{4.23}$$

$$\delta^{O}C_{pc} = {}^{O}C_{pc}-{}^{O}\bar{C}_{pc} \approx a*(x-L/2) \tag{4.24}$$

Diese lineare Näherung ist jedoch nur für geringe Diffusionen geeignet. Zudem ist der Parameter a eine flussabhängige Größe, wodurch diese Näherung bei der folgenden Flussmodulation kritisch zu betrachten ist.

Mit dieser Näherung können nun folgende Umformungen durchgeführt werden:

$$\frac{1}{L}*\int_0^L \frac{{}^{O}C_{pc}}{{}^{O}C_{pc}+K_m}dx = \frac{{}^{O}\bar{C}_{pc}}{{}^{O}\bar{C}_{pc}+K_m}$$

$$+\frac{1}{L}*\int_0^L \frac{K_m}{({}^O\bar{C}_{pc}+K_m)^2}*\delta^O C_{pc}dx$$

$$+\frac{1}{L}\sum_{i=2}^{\infty}\int_0^L(-1)^{i+1}*\frac{K_m}{({}^O\bar{C}_{pc}+K_m)^{i+1}}*\delta^O C^i_{pc}dx \quad (4.25)$$

$$\overset{4.20}{\approx} \frac{{}^O\bar{C}_{pc}}{{}^O\bar{C}_{pc}+K_m}$$

$$+\frac{1}{L}\sum_{i=2}^{\infty}((-1)^{i+1}*\frac{K_m}{({}^O\bar{C}_{pc}+K_m)^{i+1}}$$

$$*\int_0^L(ax-aL/2)^i dx) \quad (4.26)$$

$$= \frac{{}^O\bar{C}_{pc}}{{}^O\bar{C}_{pc}+K_m}$$

$$+\frac{1}{L}\sum_{i=2}^{\infty}((-1)^{i+1}*\frac{K_m}{({}^O\bar{C}_{pc}+K_m)^{i+1}}*\int_{-a*L/2}^{a*L/2}c^i\frac{dc}{a}) \quad (4.27)$$

$$= \frac{{}^O\bar{C}_{pc}}{{}^O\bar{C}_{pc}+K_m}$$

$$+\frac{1}{a*L}\sum_{i=2}^{\infty}((-1)^{i+1}*\frac{K_m}{({}^O\bar{C}_{pc}+K_m)^{i+1}}$$

$$*(a*L/2)^{i+1}*(1-(-1)^{i+1})*\frac{1}{2i+1} \quad (4.28)$$

$$\overset{j=2i}{=} \frac{{}^O\bar{C}_{pc}}{{}^O\bar{C}_{pc}+K_m}$$

$$+\frac{1}{a*L}\sum_{j=1}^{\infty}(-1)^{2j+1}*\frac{K_m}{({}^O\bar{C}_{pc}+K_m)^{2j+1}}$$

$$*(a*L/2)^{2j+1}*(1-(-1)^{2j+1})*\frac{1}{2j+1} \quad (4.29)$$

$$= \frac{{}^O\bar{C}_{pc}}{{}^O\bar{C}_{pc}+K_m}$$

$$-\frac{K_m}{{}^O\bar{C}_{pc}+K_m}*\sum_{j=1}^{\infty}*(\frac{a*L/2}{{}^O\bar{C}_{pc}+K_m})^{2j}*\frac{1}{2j+1} \quad (4.30)$$

$$= \frac{{}^O\bar{C}_{pc}}{{}^O\bar{C}_{pc}+K_m}$$

$$-\frac{K_m}{{}^O\bar{C}_{pc}+K_m}*\frac{artanh(\omega)-\omega}{\omega} \quad (4.31)$$

mit

$$\omega = \frac{a*L}{2*({}^O\bar{C}_{pc}+K_m)} \quad (4.32)$$

$$= -\frac{v_{max}}{2 * F_{pl} * W_{pl} * ({}^{O}\bar{C}_{pc} + K_m)} \tag{4.33}$$

Das System ergibt sich wie folgt:

$$\frac{d}{dt}{}^{O}\bar{C}_{pl} = \frac{F_{pl}}{V_{pl}}({}^{O}C_{pl_{in}} - {}^{O}C_{pl_L}) - \frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}({}^{O}\bar{C}_{pl} - {}^{O}\bar{C}_{isf}) \tag{4.34}$$

$$\frac{d}{dt}{}^{O}\bar{C}_{pl} = \frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}({}^{O}\bar{C}_{pl} - {}^{O}\bar{C}_{isf}) - \frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{isf} - {}^{O}\bar{C}_{pc}) \tag{4.35}$$

$$\begin{aligned}\frac{d}{dt}({}^{O}\bar{C}_{pc} + \bar{C}_{MbO_2}) = {} & \frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{isf} - {}^{O}\bar{C}_{pc}) - \frac{v_{max}}{V_{pc} * W_{pc}}(\frac{{}^{O}\bar{C}_{pc}}{{}^{O}\bar{C}_{pc} + K_m} \\ & - \frac{K_m}{{}^{O}\bar{C}_{pc} + K_m} * \frac{artanh(\omega) - \omega}{\omega})\end{aligned} \tag{4.36}$$

Es gilt weiterhin:

$$C_{MbO_2} = C_{Mbpc} * \frac{K_{MbO_2} * {}^{O}C_{pc}}{1 + K_{MbO_2} * {}^{O}C_{pc}} \tag{4.37}$$

$$= C_{Mbpc} * \frac{{}^{O}C_{pc}}{{}^{O}C_{pc} + \alpha_{O_2} * P_{50}^{Mb}} \tag{4.38}$$

In Analogie zur Herleitung für den Stoffumsatzterm ergibt sich damit:

$$\bar{C}_{MbO_2} = \frac{1}{L}\int_0^L C_{MbO_2}dx \tag{4.39}$$

$$= C_{Mbpc}(\frac{{}^{O}\bar{C}_{pc}}{{}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb}} - \frac{\alpha_{O_2} * P_{50}^{Mb}}{{}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb}} * \frac{artanh(\sigma) - \sigma}{\sigma}) \tag{4.40}$$

mit

$$\sigma = \frac{a * L}{2 * ({}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb})} \tag{4.41}$$

$$= -\frac{v_{max}}{2 * F_{pl} * W_{pl} * ({}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb})} \tag{4.42}$$

Daraus folgt:

$$\begin{aligned}\frac{d}{dt}\bar{C}_{MbO_2} \approx {} & \frac{d}{dt}(C_{Mbpc} * (\frac{{}^{O}\bar{C}_{pc}}{{}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb}} \\ & - \frac{\alpha_{O_2} * P_{50}^{Mb}}{{}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb}} * \frac{artanh(\sigma) - \sigma}{\sigma}))\end{aligned} \tag{4.43}$$

$$
\begin{aligned}
&= C_{Mbpc} * \Big(\frac{d}{dt}\frac{{}^{O}\bar{C}_{pc}}{{}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb}} \\
&\quad + \frac{d}{dt}\frac{\alpha_{O_2} * P_{50}^{Mb}}{{}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb}} \\
&\quad - \frac{d}{dt}\frac{\alpha_{O_2} * P_{50}^{Mb} * 2}{a * L} * arctanh(\sigma)\Big) \qquad (4.44) \\
&= C_{Mbpc} * \frac{\alpha_{O_2} * P_{50}^{Mb}}{({}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb})^2 - (a * L/2)^2} * \frac{d}{dt}\,{}^{O}\bar{C}_{pc} \qquad (4.45)
\end{aligned}
$$

Ergibt sich die letzte der Gleichungen zu:

$$
\begin{aligned}
(1 + \frac{C_{Mbpc} * \alpha_{O_2} * P_{50}^{Mb}}{({}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb})^2 - (a * L/2)^2})\frac{d}{dt}{}^{O}\bar{C}_{pc} \quad &= \\
\frac{{}^{O}PS_{pc}}{V_{isf} W_{isf}}({}^{O}\bar{C}_{isf} - {}^{O}\bar{C}_{pc}) & \\
- \frac{v_{max}}{V_{pc} * W_{pc}}(\frac{{}^{O}\bar{C}_{pc}}{{}^{O}\bar{C}_{pc} + K_m} - \frac{K_m}{{}^{O}\bar{C}_{pc} + K_m} * \frac{artanh(\omega) - \omega}{\omega}) & \qquad (4.46)
\end{aligned}
$$

Womit ersichtlich ist, dass die Berücksichtigung der Bindung von Sauerstoff an Myoglobin sich im effektiven Stirred Tank-Modell ausschließlich in einer Verlangsamung der Prozesse im Parenchymalzellkompartiment niederschlägt.

In dem Gleichungssystem muss nun noch die Ausflusskonzentration durch einen entsprechenden Term in Abhängigkeit der Durchschnittskonzentration der Kapillare ersetzt werden:

$$
\begin{aligned}
\frac{F_{pl}}{V_{pl}} * ({}^{O}C_{pl_{in}} - {}^{O}C_{pl_L}) &\approx \frac{F_{pl}}{V_{pl}} * (b - a * L - b) \qquad (4.47) \\
&= \frac{F_{pl}}{V_{pl}}(2b - 2 * (a * L/2 + b)) \qquad (4.48) \\
&\approx \frac{2 * F_{pl}}{V_{pl}}({}^{O}C_{pl_{in}} - {}^{O}\bar{C}_{pl}) \qquad (4.49)
\end{aligned}
$$

Im Vergleich zu einem direkt aufgestellten Stirred-Tank-Modell, kann durch diese Herleitung die Verlangsamung der Reaktionen in den Parenchymalzellen berücksichtigt werden. Zudem ergibt sich im Flussterm ein zusätzlicher Vorfaktor 2. Demzufolge müsste bei der Überführung vom Krogh-Zylinder-Modell zum Stirred-Tank-Modell der effektive Fluss näherungsweise verdoppelt werden. Diesen Überführungsfaktor wird nachfolgend mit k_s bezeichnet, wobei s durch das jeweilige Substrat zu ersetzen ist, und wird in Kapitel 4.3 genauer betrachtet.

In den Gleichungen für CO_2,HCO_3^-, H^+, $H_2PO_4^-$ und HPO_4^{2-} ist der einzige nichtlineare Term, der somit also nicht direkt überführt werden kann der Term des Massenwirkungsgesetzes für die Pufferwirkungen. Dieser geht wie folgt vom Krogh-Zylinder-Modell in das Stirred Tank Modell über (exemplarisch für den Carbonatpuffer. Herleitung des Phosphatpuffers erfolgt analog):

$$\begin{aligned} K_r(kf^C C_r - kb^b C_r{}^h C_r) &= K_r(kf({}^C\bar{C}_r + \delta^C C_r) \qquad (4.50)\\ &\quad -kb({}^b\bar{C}_r + \delta^b C_r)({}^h\bar{C}_r + \delta^h C_r)) \\ &= K_r * (kf * ({}^C\bar{C}_r + \delta^C C_r) - kb * ({}^b\bar{C}_r *{}^h \bar{C}_r \\ &\quad +{}^b\bar{C}_r * \delta^h C_r +{}^h \bar{C}_r * \delta^b C_r + \delta^b C_r * \delta^h C_r)) \qquad (4.51) \end{aligned}$$

Damit folgt:

$$\begin{aligned} &\frac{1}{L}\int_0^L K_r * (kf *{}^C C_r - kb *{}^b C_r *{}^h C_r)dx \qquad (4.52)\\ = \;& \frac{1}{L}\int_0^L K_r * (kf * ({}^C\bar{C}_r + \delta^C C_r) - kb * (\frac{1}{L}\int_0^L {}^b\bar{C}_r *{}^h \bar{C}_r dx \\ &+\frac{1}{L}\int_0^L {}^b\bar{C}_r * \delta^h C_r dx + \frac{1}{L}\int_0^L {}^h\bar{C}_r * \delta^b C_r dx + \frac{1}{L}\int_0^L \delta^b C_r * \delta^h C_r dx)) \qquad (4.53)\\ = \;& K_r * (kf *{}^C \bar{C}_r - kb *{}^b \bar{C}_r *{}^h \bar{C}_r) - K_r * kb * \frac{1}{L}\int_0^L \delta^b C_r * \delta^h C_r dx \qquad (4.54) \end{aligned}$$

Da der pH-Wert und somit auch die Konzentration von H^+ gemäß den bisherigen Simulationsergebnissen als axial nahezu konstant angenommen werden kann, gilt somit:

$$\delta^h C_r \approx 0 \qquad (4.55)$$

Damit vereinfacht sich die Überführung wie folgt:

$$\frac{1}{L}\int_0^L K_r(kf *{}^C C_r - kb *{}^b C_r *{}^h C_r)dx \approx K_r(kf *{}^C \bar{C}_r - kb *{}^b \bar{C}_r *{}^h \bar{C}_r) \qquad (4.56)$$

Selbiges ergibt sich für die Terme des Phosphatpuffers sowie der Gleichungen für H^+ wo im Vorfaktor der Pufferwirkung die Konzentration ebenfalls als axial konstant angenommen werden kann.

Insgesamt lautet damit das hergeleitete System gewöhnlicher Differentialgleichungen

wie folgt:

$$\frac{d^{O}\bar{C}_{pl}}{dt} = \frac{k_{O_2} * F_{pl}}{V_{pl}}({}^{O}C_{pl_{in}} - {}^{O}\bar{C}_{pl}) - \frac{{}^{O}PS_{cap}}{V_{pl}W_{pl}}({}^{O}\bar{C}_{pl} - {}^{O}\bar{C}_{isf}) \tag{4.57}$$

$$\frac{d^{O}\bar{C}_{isf}}{dt} = \frac{{}^{O}PS_{cap}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{pl} - {}^{O}\bar{C}_{isf}) - \frac{{}^{O}PS_{pc}}{V_{isf}W_{isf}}({}^{O}\bar{C}_{isf} - {}^{O}\bar{C}_{pc}) \tag{4.58}$$

$$\frac{d^{O}\bar{C}_{pc}}{dt} = (\frac{{}^{O}PS_{pc}}{V_{pc}W_{pc}}({}^{O}\bar{C}_{isf} - {}^{O}\bar{C}_{pc}) - \frac{v_{max}}{V_{pc} * W_{pc}}(\frac{{}^{O}\bar{C}_{pc}}{{}^{O}\bar{C}_{pc} + K_m} - \frac{K_m}{{}^{O}\bar{C}_{pc} + K_m} * \frac{artanh(\omega) - \omega}{\omega})) / (1 + \frac{C_{Mbpc} * \alpha_{O_2} * P_{50}^{Mb}}{({}^{O}\bar{C}_{pc} + \alpha_{O_2} * P_{50}^{Mb})^2 - (a * L/2)^2}) \tag{4.59}$$

$$\frac{d^{C}\bar{C}_{pl}}{dt} = \frac{k_{CO_2} * F_{pl}}{V_{pl}}({}^{C}C_{pl_{in}} - {}^{C}\bar{C}_{pl}) - \frac{{}^{C}PS_{cap}}{V_{pl}W_{pl}}({}^{C}\bar{C}_{pl} - {}^{C}\bar{C}_{isf}) + K_{pl}(kb_1^b\bar{C}_{pl}^h\bar{C}_{pl} - kf_1^C\bar{C}_{pl}) \tag{4.60}$$

$$\frac{d^{C}\bar{C}_{isf}}{dt} = \frac{{}^{C}PS_{cap}}{V_{isf}W_{isf}}({}^{C}\bar{C}_{pl} - {}^{C}\bar{C}_{isf}) - \frac{{}^{C}PS_{pc}}{V_{isf}W_{isf}}({}^{C}\bar{C}_{isf} - {}^{C}\bar{C}_{pc}) + K_{isf}(kb_1^b\bar{C}_{isf}^h\bar{C}_{isf} - kf_1^C\bar{C}_{isf}) \tag{4.61}$$

$$\frac{d^{C}\bar{C}_{pc}}{dt} = \frac{{}^{C}PS_{pc}}{V_{pc}W_{pc}}({}^{C}\bar{C}_{isf} - {}^{C}\bar{C}_{pc}) + RQ * \frac{v_{max}}{V_{pc} * W_{pc}}(\frac{{}^{O}\bar{C}_{pc}}{{}^{O}\bar{C}_{pc} + K_m} - \frac{K_m}{{}^{O}\bar{C}_{pc} + K_m} * \frac{artanh(\omega) - \omega}{\omega}) + K_{pc}(kb_1^b\bar{C}_{pc}^h\bar{C}_{pc} - kf_1^C\bar{C}_{pc}) \tag{4.62}$$

$$\frac{d^{b}\bar{C}_{pl}}{dt} = \frac{k_{HCO_3^-} * F_{pl}}{V_{pl}}({}^{b}C_{pl_{in}} - {}^{b}\bar{C}_{pl}) - \frac{{}^{b}PS_{cap}}{V_{pl}W_{pl}}(R_{cap}^b\bar{C}_{pl} - {}^{b}\bar{C}_{isf}) - K_{pl}(kb_1^b\bar{C}_{pl}^h\bar{C}_{pl} - kf_1^C\bar{C}_{pl}) \tag{4.63}$$

$$\frac{d^{b}\bar{C}_{isf}}{dt} = \frac{{}^{b}PS_{cap}}{V_{isf}W_{isf}}(R_{cap}^b\bar{C}_{pl} - {}^{b}\bar{C}_{isf}) - \frac{{}^{b}PS_{pc}}{V_{isf}W_{isf}}(R_{pc}^b\bar{C}_{isf} - {}^{b}\bar{C}_{pc}) - K_{isf}(kb_1^b\bar{C}_{isf}^h\bar{C}_{isf} - kf_1^C\bar{C}_{isf}) \tag{4.64}$$

$$\frac{d^{b}\bar{C}_{pc}}{dt} = \frac{{}^{b}PS_{pc}}{V_{pc}W_{pc}}(R_{pc}^b\bar{C}_{isf} - {}^{b}\bar{C}_{pc}) - K_{pc}(kb_1^b\bar{C}_{pc}^h\bar{C}_{pc} - kf_1^C\bar{C}_{pc}) \tag{4.65}$$

$$\frac{d^{B}\bar{C}_{pl}}{dt} = \frac{k_{HPO_4^{2-}} * F_{pl}}{V_{pl}}({}^{B}C_{pl_{in}} - {}^{B}\bar{C}_{pl}) - \frac{{}^{B}PS_{cap}}{V_{pl}W_{pl}}(R_{cap}^2 * {}^{B}\bar{C}_{pl} - {}^{B}\bar{C}_{isf}) - (kb_2^B\bar{C}_{pl}^h\bar{C}_{pl} - kf_2^H\bar{C}_{pl}) \tag{4.66}$$

$$\frac{d^B\bar{C}_{isf}}{dt} = \frac{^BPS_{cap}}{V_{isf}W_{isf}}(R^2_{cap} *^B \bar{C}_{pl} -^B \bar{C}_{isf}) - (kb_2^B \bar{C}^h_{isf}\bar{C}_{isf} - kf_2^H \bar{C}_{isf}) \quad (4.67)$$

$$\frac{d^B\bar{C}_{pc}}{dt} = (kb_2^B \bar{C}^h_{pc}\bar{C}_{pc} - kf_2^H \bar{C}_{pc}) \quad (4.68)$$

$$\frac{d^H\bar{C}_{pl}}{dt} = \frac{k_{H_2PO_4^-} * F_{pl}}{V_{pl}}(^HC_{pl_{in}} -^H \bar{C}_{pl}) - \frac{^HPS_{cap}}{V_{pl}W_{pl}}(R^H_{cap}\bar{C}_{pl} -^H \bar{C}_{isf}) + (kb_2^B \bar{C}^h_{pl}\bar{C}_{pl} - kf_2^H \bar{C}_{pl}) \quad (4.69)$$

$$\frac{d^H\bar{C}_{isf}}{dt} = \frac{^HPS_{cap}}{V_{isf}W_{isf}}(R^H_{cap}\bar{C}_{pl} -^H \bar{C}_{isf}) + (kb_2^B \bar{C}^h_{isf}\bar{C}_{isf} - kf_2^H \bar{C}_{isf}) \quad (4.70)$$

$$\frac{d^H\bar{C}_{pc}}{dt} = (kb_2^B \bar{C}^h_{pc}\bar{C}_{pc} - kf_2^H \bar{C}_{pc}) \quad (4.71)$$

$$\frac{d^hC_{pl}}{dt} = \frac{k_{H^+} * F_{pl}}{V_{pl}}(^hC_{pl_{in}} -^h \bar{C}_{pl}) - \frac{^hPS_{cap}}{V_{pl}W_{pl}}(^h\bar{C}_{pl} - R^h_{cap}\bar{C}_{isf}) - \frac{2,303^h}{\beta_{pl}}\bar{C}_{pl}K_{pl}(kb_1^b \bar{C}^h_{pl}\bar{C}_{pl} - kf_1^C \bar{C}_{pl}) - \frac{2,303^h}{\beta_{pl}}\bar{C}_{pl}(kb_2^B \bar{C}^h_{pl}\bar{C}_{pl} - kf_2^H \bar{C}_{pl}) \quad (4.72)$$

$$\frac{d^h\bar{C}_{isf}}{dt} = \frac{^hPS_{cap}}{V_{isf}W_{isf}}(^h\bar{C}_{pl} - R^h_{cap}\bar{C}_{isf}) - \frac{^hPS_{pc}}{V_{isf}W_{isf}}(^h\bar{C}_{isf} - R^h_{pc}\bar{C}_{pc}) - \frac{2,303^h}{\beta_{isf}}\bar{C}_{isf}K_{isf}(kb_1^b \bar{C}^h_{isf}\bar{C}_{isf} - kf_1^C \bar{C}_{isf}) - \frac{2,303^h}{\beta_{isf}}\bar{C}_{isf}(kb_2^B \bar{C}^h_{isf}\bar{C}_{isf} - kf_2^H \bar{C}_{isf}) \quad (4.73)$$

$$\frac{d^h\bar{C}_{pc}}{dt} = \frac{^hPS_{pc}}{V_{pc}W_{pc}}(^h\bar{C}_{isf} - R^h_{pc}\bar{C}_{pc}) - \frac{2,303^h}{\beta_{pc}}\bar{C}_{pc}K_{pc}(kb_1^b \bar{C}^h_{pc}\bar{C}_{pc} - kf_1^C \bar{C}_{pc}) - \frac{2,303^h}{\beta_{pc}}\bar{C}_{pc}(kb_2^B \bar{C}^h_{pc}\bar{C}_{pc} - kf_2^H \bar{C}_{pc}) \quad (4.74)$$

Insgesamt ergibt sich im Vergleich zu einem theoretisch hergeleitetem Stirred-Tank-Modell ein recht ähnliches System von gewöhnlichen Differentialgleichungen. Unterschiede gibt es bei der mathematischen Beschreibung des konvektiven Flusses, bei welcher hier ein zusätzlicher Überführungsfaktor zum tragen kommt. Somit ist der Fluss in einem regulären Stirred-Tank-Modell unterrepräsentiert. Zudem kann die Bindung an Myoglobin über den entsprechenden Korrekturterm der Differentialgleichung für Sauerstoff in den Parenchymzellen berücksichtigt werden. Für den Gulosity-Term konnte

ebenfalls ein Korrekturterm bestimmt werden. Dieser wird aber aufgrund seines minimalen Einflusses in der Implementierung des Stirred-Tank-Modelles vernachlässigt und dient hier nur der Fehlerabschätzung.

4.3 Einfluss und Wahl des Überführungsfaktors

Bei der Überführung des Krogh-Zylinder-Modells in ein einfacheres Stirred-Tank-Modell wurde bei der Überführung des Flusstermes ein zusätzlicher Saklierungsfaktor k_s eingefügt, um den Übergang der Ausflusskonzentration auf die durchschnittliche Gefäßkonzentration zu gewährleisten. Im stationären Zustand kann dieser als Maß der Linearität des Systems angesehen werden und ist konstant und durch die numerische Lösung des Differentialgleichungssystems bestimmbar. Bei der Systemdynamik durch die einsetzende hyperkapnische Reaktion beeinflusst der Überführungsfaktor jedoch die Systemdynamik und ist somit exakterweise als zeitabhängige Größe aufzufassen, die wie folgt definiert ist:

$$k_s(t) = \frac{{}^sC_{in}(t) - {}^sC_{pl}(t,L)}{{}^sC_{in}(t) - {}^s\bar{C}_{pl}(t)} \tag{4.75}$$

Über den Überführungsfaktor lässt sich die mittlere Gefäßkonzentraton des Stirred-Tank-Modells dann auch wieder in eine Ausflusskonzentration überführen:

$${}^sC_{pl}(t,L) = (1-k_s(t)) * {}^sC_{in}(t) + k_s(t) * {}^s\bar{C}_{pl}(t) \tag{4.76}$$

Jedoch ist es nicht möglich, den Überführungsfaktor aus dem Stirred-Tank-Modell abzuleiten - er ergibt sich ausschließlich aus der Berechnung des vollständigen Krogh-Zylinder-Modells. Da das Stirred-Tank-Modell jedoch hergeleitet wurde, um nicht das vollständige Krogh-Zylinder-Modell lösen zu müssen, ist nun die Frage zu klären, inwieweit eine Variation der Parameter zu einer Verschiebung der Überführungsfaktoren führt, bzw. inwieweit sich diese Verschiebung auf die Resultate in der Systemdynamik auswirken.

Dazu wurden die Überführungsfaktoren aus dem Krogh-Zylinder-Modell mit Standardparametern übernommen und anschließend Simulationen unter Variation einzelner Parameter durchgeführt und die Ergebnisse des Krogh-Zylinder-Modells mit denen des Stirred-Tank-Modells verglichen.

Dabei ist festzustellen, dass für die meisten Parameter eine Variation ohne größere

Änderung der Simulationsergebnisse möglich ist, es in der Regel sogar hinreichend für die Systemdynamik ist, wenn die Überführungsparameter der beiden stationären Zustände verwendet werden (s. Tabelle 4.1).

Tabelle 4.1: Auswirkung der Paramervariation

Parameter	**Min**	**Standard**	**Max**	**Auswirkung**
V_{pl}	0.02	0.07	0.2	+
V_{isf}	0.05	0.2	0.5	+
V_{pc}	0.2	0.7	2	+
W_{pl}	0.8	0.94	1	+
W_{isf}	0.8	0.92	1	+
W_{pc}	0.7	0.8	1	Einstellphase leicht träger
RQ	0.7	0.8	1	+
K_m	$2*10^{-7}$	$7*10^{-7}$	$2*10^{-6}$	+
v_{max}	$1.5*10^{-6}$	$2.5*10^{-6}$	$4*10^{-6}$	+
${}^{O}PS_{cap}$	20	200	2000	+
${}^{O}PS_{pc}$	200	2000	20000	falsche Vorhersage!
${}^{C}PS_{cap}$	20	200	2000	+
${}^{C}PS_{pc}$	200	2000	20000	falsche Vorhersage!
${}^{h}PS_{cap}$	2	20	200	+
${}^{h}PS_{pc}$	20	200	2000	+
${}^{b}PS_{cap}$	2	20	200	+
${}^{b}PS_{pc}$	20	200	2000	+
${}^{H}PS_{cap}$	2	20	200	+
${}^{B}PS_{pc}$	2	20	200	+
R_{cap}	0.5	0.63	1	+
R_{pc}	0.6	0.79	1	+
β_{pl}	3	6	12	+
β_{isf}	12	24	48	+
β_{pc}	23	45	90	+

Tabelle 4.1 – *Fortsetzung*

Parameter	**Min**	**Standard**	**Max**	**Auswirkung**
K_{pl}	10	100	1000	+
K_{isf}	500	5000	50000	+
K_{pc}	1000	10000	100000	+
kf_1	0.012	0.12	1.2	+
kf_2	0.012	0.12	1.2	+

Einheit der Parameter entspricht jeweils der im Modell beschriebenen Einheit

Somit kann das Stirred-Tank-Modell für die Variation der meisten Parameter Anwendung finden, wobei die so erhaltenen Resultate am Ende stets gegen das vollständige Krogh-Zylinder-Modell validiert werden sollten. Für die Simulationen wurde stets einmal das Krogh-Zylinder-Modell durchgeführt und die dort erhaltenen Parameter, welche in der Regel zwischen 1,6 und 2 liegen, als Ausgangsbasis für das Stirred-Tank-Modell verwendet.

5 Bayes'sche Datenanalyse und Metropolisalgorithmus

Die Bayes'sche Datenanalyse beruht auf der Verarbeitung von Wahrscheinlichkeiten basierend auf der Anwendung des Bayes'schen Theorems:

$$P(\theta|data) \quad = \quad \frac{P(data|\theta) * P(\theta)}{P(data)} \tag{5.1}$$

Dies kann man als Lernprozess auffassen, indem aus einer vorliegenden a-prio-Wahrscheinlichkeitsverteilung $P(\theta)$ der M Parameter $\theta = \theta_1 ... \theta_M$ unter Berücksichtigung der Likelihood der Daten $P(data|\theta)$ die a-posterio-Verteilung $P(\theta|data)$ hergeleitet wird. Dabei müssen jedoch bestimmte Annahmen über die Wahrscheinlichkeitsverteilungen getroffen werden [11] [15].

Für die a-priori-Wahrscheinlichkeit $P(\theta)$ wird dabei, da alle Parameter positiv sind und zudem Näherungswerte aus anderen Modellen vorhanden sind, eine Exponentialverteilung zu Grunde gelegt.

$$P(\theta) \sim \quad \begin{cases} \exp(-\frac{\theta}{\theta_0}) & , \theta > 0 \\ 0 & , sonst \end{cases} \tag{5.2}$$

wobei θ_0 den aus anderen Modellen übernommenen Erwartungswert darstellt. Die exponentielle Verteilung ergibt sich dabei direkt aus dem MaxEnt-Prinzip unter der Nebenbedingung bekannter Erwartungswerte. Durch das MaxEnt-Prinzip ist sichergestellt, dass nicht mehr Vorinformation im Modell verwendet wird, als tatsächlich vorhanden ist [9] [11].

Die Wahrscheinlichkeit $P(data|\theta)$ resultiert aus der Betrachtung der Messfehler:

$$messdata_i \quad = \quad realdata_i + \varepsilon_i \tag{5.3}$$

wobei ε_i der jeweilige Messfehler ist. Dieser Fehler ist unkorreliert und normalverteilt mit einer Standardabweichung, die sich aus der Genauigkeit der Messapparatur ergibt. Somit gilt nach Maximum-Likelihood-Prinzip:

$$P(data|\theta) \quad \sim \quad \exp(-\frac{(data - modell(\theta))^2}{2 * \sigma^2}) \tag{5.4}$$

Um aus dieser hochdimensionalen Wahrscheinlichkeitsverteilung nun einzelne Parame-

ter zu bestimmen, muss folgendes hochdimensionales Integral gelöst werden [11]:

$$< \theta_k > \quad = \quad \frac{1}{N} \int \theta_k * P(\theta|data) d\theta \tag{5.5}$$

Die direkte Berechnung dieses Integral kann umgangen werden, indem stattdessen der Metropolis-Algorithmus verwendet wird, dessen Parameterverteilung gegen die Integrallösung konvergiert [15].

Die Akzeptanzwahrscheinlichkeit eines einzelnen Schrittes hägt im Metropolisalgorithmus dabei nur von dem Metropolisverhältnis ab

$$r \quad = \quad \frac{P(\theta_2|data)}{P(\theta_1|data)} \tag{5.6}$$

$$\quad = \quad \frac{P(data|\theta_2) * P(\theta_2)}{P(data|\theta_1) * P(\theta_1)} \tag{5.7}$$

und ist damit sogar unabhängig von P(data), sodass für P(data) keine Annahmen getroffen werden müssen. Die Normierung der Wahrscheinlichkeiten auf 1 ist irrelevant, da nur das Metropolisverhältnis benötigt wird und sich die Normierung darin ebenfalls herauskürzt.

Der Fehler für die Parameter wird über die Varianz bestimmt und kann aus der im Metropolisalgorithmus bestimmten Parameterfolge effektiv geschätzt werden.

5.1 Vorbetrachtungen

Zunächst wurde untersucht, welche der Parameter sich bei gegebenen Messdaten überhaupt exakt bestimmen lassen. Dazu wurden Modellergebnisse mit Messfehlern behaftet und als experimentelle Vergleichsdaten verwendet.

Dabei wurden folgende Annahmen für den Versuchsaufbau getroffen:

- die Ausflusskonzentrationen werden aller 10 Sekunden gemessen
- jeder so erhaltene Messwert ist mit einem normalverteilten Fehler behaftet
- diese Messfehler sind unabhängig voneinander
- die Perfusion findet mit konstantem Fluss statt

Als Messfehler σ wurden dabei für Sauerstoff und Kohlenstoffdioxid 0.1 mmHg, für den pH-Wert 0.01 und für die Bicarbonatkonzentration 0.1 mM angesetzt.

Im Metropolisalgorithmus wird der neue Parameter zufällig gleichverteilt aus einem Intervall gewählt, welches vom aktuellen Parameter und der Schrittweite SW abhängig ist.

Für eine optimale Konvergenz des Metropolisalgorithmus ist die Schrittweite dabei so zu wählen, dass die Akzeptanzrate neuer Parameter bei ca. 35% liegt [9]. Dafür werden zunächst Warm-Up-Zyklen durchgeführt, in denen die Schrittweite angepasst wird, bis die Akzeptanzrate ca. 35% beträgt.

Für die Stabilität des Modelles und der Überführungsfaktoren wurden die Parameter zudem auf das 0.1 bis 10-fache des Ursprünglichen Parameterwertes eingeschränkt.

Aufgrund der Rechenzeit des Metropolisalgorithmus wurde die Testvorhersage dabei wie folgt eingeschränkt: Für die Parameter V_{pl}, W_{pl}, VW_{isf}, VW_{pc}, ${}^{O}PS_{cap}$, ${}^{O}PS_{pc}$, α_{O_2}, K_m, v_{max}, C_{Mb} und P_{Mb}^{50} wurde eine gemeinsame Variation durchgeführt, welche nur gegen die Messkurve von O_2 validiert wird. Für die Parameter β_{pl}, kf_2 und V_{pl} wurde jeweils eine einzelne Parametervaraition gegen alle 4 Messkurven (O_2, CO_2, HCO_3^- und H^+) der Ausflusskonzentration durchgeführt.

Für die Genauigkeit der Parametervorhersagen wurde als Maß der Variationskoeffizient ρ herangezogen, also das Verhältnis zwischen Standardabweichung σ_k und Erwartungswert $< \theta_k >$.

$$\rho_k = \frac{\sigma_k}{< \theta_k >} \tag{5.8}$$

Tabelle 5.1: Genauigkeit der Parameterbestimmung

Parameter	ρ	**Parameter**	ρ
V_{pl}	0.1207	W_{pl}	0.0012
VW_{isf}	0.0447	VW_{pc}	0.0192
${}^{O}PS_{cap}$	0.2168	${}^{O}PS_{pc}$	0.2473
v_{max}	0.0012	K_m	0.1402
C_{Mb}	0.4587	P_{Mb}^{50}	0.4069
α_{O_2}	0.0012		
V_{pl}	0.0314	β_{pl}	0.0050
kf_2	0.1030		

Es ist festzustellen, dass sich die meisten Parameter, welche für dieses Modell simuliert wurden, relativ genau bestimmen lassen. Lediglich die Membranpermeabilitäten und die Parameter für Myoglobin lassen sich aus den Sauerstoffdaten nur relativ ungenau

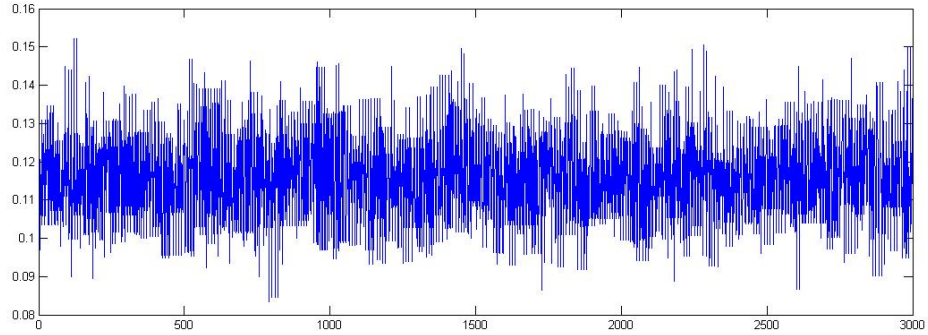

Abbildung 5.1: Verlauf des Parameterwertes für kf_2 in den einzelnen Schritten des Metropolisalgorithmus

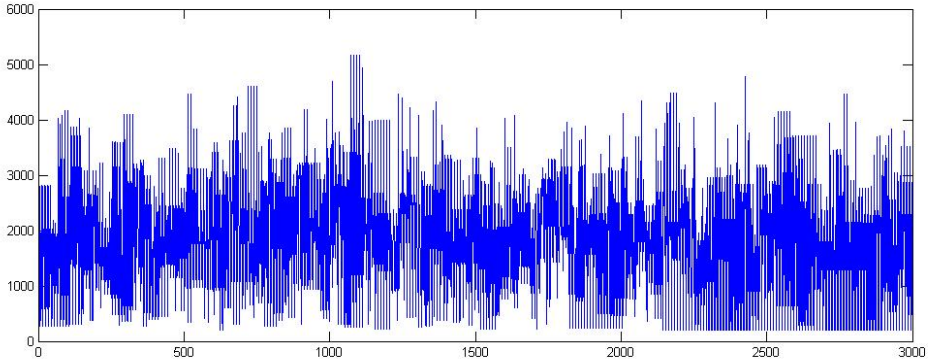

Abbildung 5.2: Verlauf des Parameterwertes für ${}^{O}PS_{cap}$ in den einzelnen Schritten des Metropolisalgorithmus. Die untere Schranke von 200, die mehrfach erreicht wird, ist gegeben durch die Einchränkung auf 10% des ursprünglichen Parameterwertes. Da dieser Wert erreicht wird, können alle Aussagen des Stirred-Tank-Modells nicht als verlässlich angenommen werden (s. Kapitel 4.3)

ermitteln. Dies liegt in deren geringem Einfluss auf die Ausflusskonzentration begründet. Zudem ist am Beispiel des Plasmavolumens festzustellen, dass durch zusätzliche Datenpunkte und die Variation von weniger Parameter die Vorraussagegenauigkeit des einzelnen Parameters steigt. Dies deutet darauf hin, dass Korrelationen zwischen den einzelnen Parametern bestehen. Zudem ist festzustellen, dass sich jene Parameter, welche sich besonders genau vorhersagen lassen, sich eine a-posteriori Verteilung ergibt, die näherungsweise normalverteilt ist.

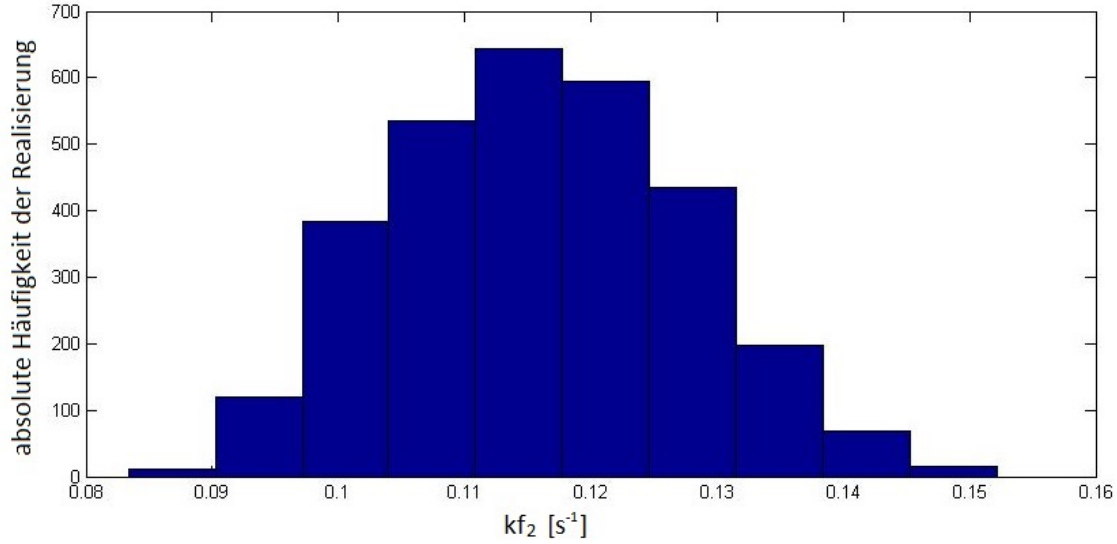

Abbildung 5.3: Häufigkeitsverteilung der Bayes'schen Datenanalyse für den Parameter kf_2

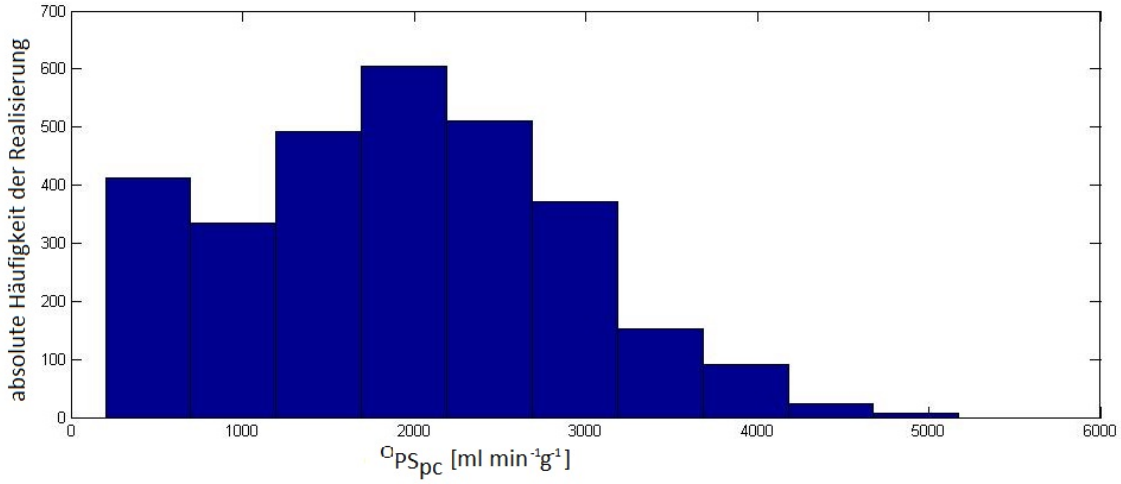

Abbildung 5.4: Häufigkeitsverteilung der Bayes'schen Datenanalyse für den Parameter ${}^{O}PS_{pc}$. Die untere Grenze der Verteilung bei 200 ist durch die zusätzliche Begrenzung auf 10% des ursprünglichen Parameterwertes gegeben.

6 Parameteroptimierung

6.1 Flussfunktion

Für die Simulation gegen tatsächliche experimentelle Datensätze ist nun die Variation des Flusses zu berücksichtigen. Dazu wird der Fluss, der von Heintz et al. [17] am Meerschweinchenherzen gemessen wurde, zunächst durch eine mathematische Formel approximiert. Empirisch wurde dabei eine Formel der Form

$$F(t) = \begin{cases} a_1 & ,t < 0 \\ \underbrace{a_1}_{basalerFluss} + \underbrace{a_2 * t^2 * \exp(-a_3 * t)}_{erste,schnelleFlussantwort} + \underbrace{max(0, a_4 - a_5 * \exp(-a_6 * t))}_{langsamere,2.Flussantwort} & ,t >= 0 \end{cases} \quad (6.1)$$

als geeignet aufgefasst. Alternativ kann diese Funktion auch wie folgt formuliert werden:

$$F(t) = \begin{cases} a_1 & ,t < 0 \\ a_1 + a_2 * t^2 * \exp(-a_3 * t) & ,0 \leq t < \tau \\ a_1 + a_2 * t^2 * \exp(-a_3 * t) + a_4 * (1 - \exp(-a_6 * (t - \tau))) & ,t \geq \tau \end{cases} \quad (6.2)$$

mit

$$\tau = \frac{1}{a_6} * \ln(\frac{a_5}{a_4}) \quad (6.3)$$

τ beschreibt dabei das Einsetzen der zweiten Flussantwort. Die erste Flussantwort erreicht bei der gegebenen Funktion ihr Maximum zum Zeitpunkt $\frac{2}{a_3}$. Mittels Bayes'scher Datenanalyse wurden nun die Parameter a_i an die experimentelle Messkurve angepasst. Da über die Parameter kein Vorwissen besteht, wird die a-priori-Wahrscheinlichkeit aller Parameter stets als konstant aufgefasst.

Tabelle 6.1: Parameter der Flussfunktion

Parameter	**Wert**	σ	**Parameter**	**Wert**	σ
$a_1[mlmin^{-1}g^{-1}]$	8.8746	0.0433	$a_2[mlmin^{-3}g^{-1}]$	20.4708	3.3641
$a_3[min^{-1}]$	2.2528	0.0808	$a_4[mlmin^{-1}g^{-1}]$	4.2172	1.0478
$a_5[mlmin^{-1}g^{-1}]$	4.8358	0.5141	$a_6[min^{-1}]$	0.2015	0.0669
$\tau[min]$	0.4391	1.0742			

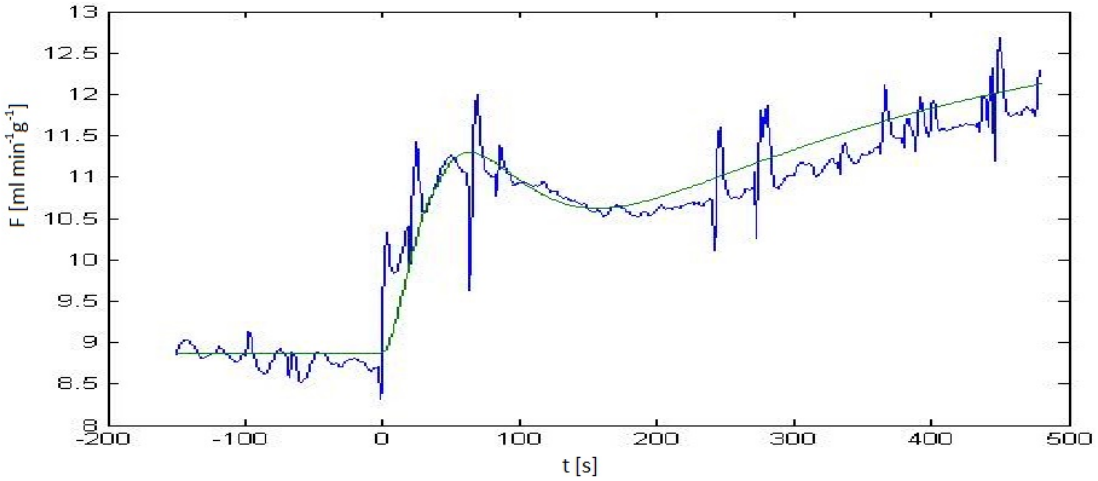

Abbildung 6.1: Experimentelle Flussfunktion und mittels Bayes'scher Datenanalyse erhaltene Approximation

Somit erreicht die erste Flussantwort ihr Maximum nach ca. 0,88 Minuten, wobei die zweite Flussantwort bereits bei ca. 0,44 Minuten einsetzt, aber erst nach dem Abfall der ersten Antwort deutlich sichtbar wird. Zudem hat τ eine extrem große Streuung und ist stark mit den anderen Parametern korrelliert - weswegen der Mittelwert für τ, welcher aus den Parametern a_i der 10x10000 Einzelschritte berechnet wurde, nicht mit dem Wert übereinstimmt, welcher sich bei Anwendung von Formel 6.3 auf die Mittelwerte der a_i ergibt. Demzufolge kann keine exakte Aussage über die Relation zwischen der Zeit des Maximums der ersten Flussantwort und dem Einsetzen der zweiten Flussantwort getroffen werden.

6.2 Parameterschätzung des Stoffumsatzes

In den Experimenten von Heintz et al. [18] am Meerschweinchenherzen wurden zeitlich hochaufgelöst die Herzrate sowie der linksventrikuläre Druck gemessen. Zwischen diesen beiden physiologischen Parametern und dem Sauerstoffverbrauch besteht näherungsweise folgender Zusammenhang:

$$G(t) \approx a + b * LVP(t) * bpm(t) \tag{6.4}$$

wobei durch Dieterich [10] folgende Werte bestimmt wurden:

$$a = 1.97 \pm 0.55 \mu molmin^{-1}g^{-1} \tag{6.5}$$

$$b = 1.47 \pm 0.48 * 10^{-4} \mu molmmHg^{-1}g^{-1} \tag{6.6}$$

Es wurde nun in Matlab unter Verwendung des Solvers für steife, gewöhnliche Differentialgleichungen 'ode15s' das Sauerstoffmodell implementiert, und ein Metropolis-

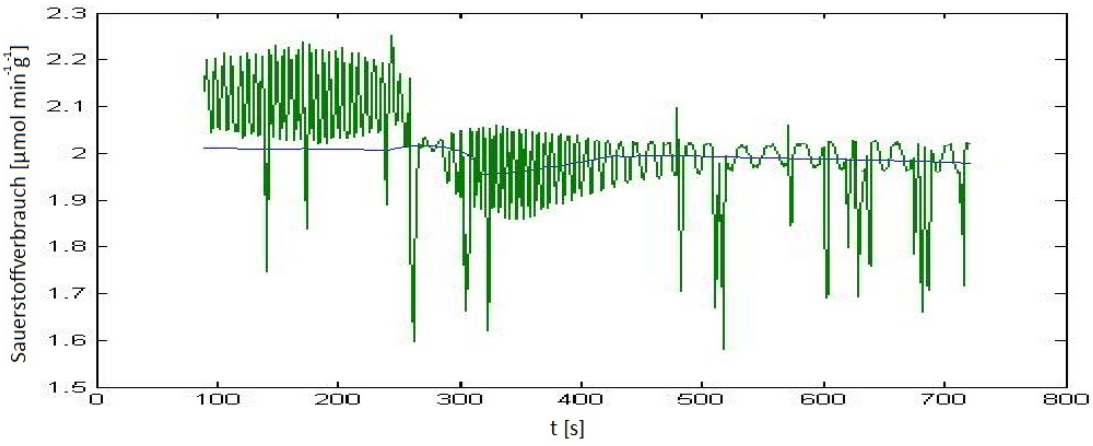

Abbildung 6.2: Näherung der experimentellen Stoffumsatzrate nach Dieterich sowie Stoffumsatzrate des Stirred-Tank-Modells bei Wahl der Parameter gemäß des Ergebnisses des Metropolis-Algorithmus

Algorithmus entwickelt, der den Gulosity-Term des Modells auswertet. Als σ wurden dabei 0.1 $\mu mol * min^{-1} * g^{-1}$ festgelegt.

Der Metropolisalgorithmus wurde nun für die Parameter ${}^{O}PS_{cap}$, ${}^{O}PS_{pc}$, K_m, v_{max}, α_{O_2}, C_{Mb} und P_{Mb}^{50} durchgeführt. Dabei ergaben sich folgende Werte:

Tabelle 6.2: Ergebnisse des Metropolisalgorithmus

Parameter	**Wert**	σ	**Parameter**	**Wert**	σ
${}^{O}PS_{cap}$	9.235	0.011	${}^{O}PS_{pc}$	1992	5.6
v_{max}	$2.0263*10^{-6}$	$0.0025*10^{-6}$	K_m	$2.53*10^{-7}$	$0.03*10^{-7}$
C_{Mb}	$4.31*10^{-5}$	$3.88*10^{-5}$	P_{Mb}^{50}	3.978	2.551
α_{O_2}	$3.477*10^{-7}$	$0.0017*10^{-7}$			

Einheit der Parameter entspricht jeweils der im Modell beschriebenen Einheit

Es ist festzustellen, dass die mit diesen Parametern erhaltene Gulosity-Funktion sich sehr gut an die experimentelle Messkurve annähert. Die meisten der Parameter konnten dabei relativ genau vorhergesagt werden, lediglich die beiden Parameter für Myoglobin weisen eine relativ große Streuung auf. Bei der Verwendung des Krogh-Zylinder-Modells mit diesen Parametern ist jedoch festzustellen, dass der Verlauf des Stoffumsatzes dort nicht mehr der experimentellen Messkurve entspricht. Dies liegt darin begründet, dass zum einen im Stirred-Tank-Modell der Korrekturterm für die Myoglobinbindung eine zusätzliche Flussabhängigkeit ins Modell einbringt, die auf Annahmen des stationären Falles beruht und im Krogh-Zylinder-Modell so nicht gegeben ist. Zum anderen

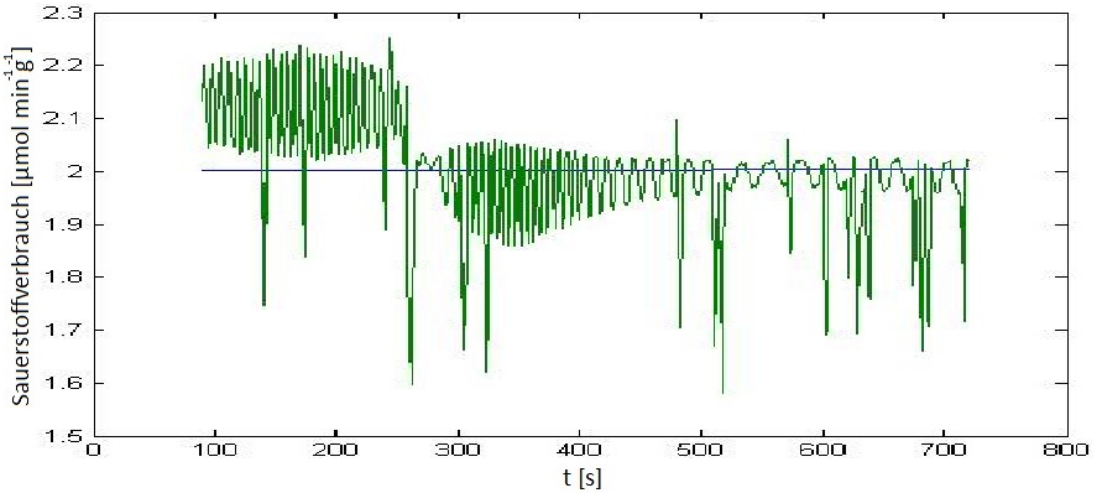

Abbildung 6.3: Näherung der experimentellen Stoffumsatzrate nach Dieterich sowie Stoffumsatzrate des Stirred-Tank-Modells bei Variation von v_{max} mittels Metropolis-Algorithmus unter Vernachlässigung der Myoglobin-Korrektur

gelten mit den hier gewählten Parametern die Vorraussetzungen für die Herleitung des Stirred-Tank-Modells nicht mehr: Durch die geringe Wahl von α_{O_2} und höhere Wahl von K_m kann im Modell nicht mehr von konstantem, vollständigem Stoffumsatz ausgegangen werden.

Es wurde nun versucht, übertragbarere Ergebnisse zu erhalten, indem der Korrekturterm für Myoglobin im Stirred-Tank vernachlässigt wird. Dadurch konnte eine Vorhersage für v_{max} vorgenommen werden.

$$v_{max} \quad = 2.02 \quad \pm 0.0148 \mu molmin^{-1}g^{-1} \tag{6.7}$$

Dabei ist jedoch festzustellen, dass die erhaltene Funktion für den Stoffumsatz kaum abhängig ist vom zu Grunde liegenden Fluss, was sich auch im Krogh-Zylinder-Modell bestätigt. Das Modell kann also die experimentelle Variation der Stoffumsatzrate bei der hyperkapnischen Reaktion, welche sich im Modell nur in der Variation des Flusses und der damit einhergehenden Änderung der O_2-Konzentration in den Parenchymalzellen niederschlägt nicht hinreichend beschreiben. Demzufolge sind die getroffenen Modellannahmen nicht geeignet, um die tatsächlichen experimentellen Vorgänge zu beschreiben und die Variation von CO_2 müsste sich im Modell auf mehr auswirken als den Fluss. Es ist denkbar, dass die Membranpermeabilitäten CO_2-, pH- und/oder flussabhängige Größen darstellen. Ebenso könnte CO_2 zusätzlich gefäßdilatierend wirken, sodass die Volumina abhängige Größen sind. CO_2 könnte sich auch in einer Form der Produktinhibierung direkt auf v_{max} auswirken. Oder die Menge des im Wasser gebundenen CO_2 wirkt sich auf die Fähigkeit von Wasser aus, O_2 zu binden und somit würde α_{O_2} eine abhängige Größe darstellen.

Für eine genauere Ursachenanalyse der Differenzen zwischen den experimentellen Daten und den Modellvorhersagen, die sich derartig auch in jSim im Krogh-Zylinder-Modell widerspiegeln und somit nicht durch die Näherungen für das Stirred-Tank-Modell bedingt sind, sind zusätzliche Experimente nötig, die den Umfang dieser Arbeit übersteigen.

7 Diskussion

Im Zuge dieser Arbeit konnte aus dem ortsabhängigen Krogh-Zylinder-Modell ein ortsunabhängiges, effektives Stirred-Tank-Modell hergeleitet werden und nachgewiesen werden, das frei aufgestellte Stirred-Tank-Modelle den Flussterm unterrepräsentieren und deshalb meist physiologisch viel zu große Volumina verwenden müssen, um diesen Effekt zu kompensieren. Dennoch besitzt auch das hier vorgestellte Stirred-Tank-Modell Einschränkungen in der Anwendbarkeit, die sich aus der eingeschränkten Anwendbarkeit der linearen Näherungen des stationären Zustandes ergeben. Bei der Untersuchung der Zeitskalen des Modells wurde festgestellt, dass die einzelnen Prozesse viel zu schnell ablaufen, als das ein einzelner davon für die auftretende Flussantwort direkt verantwortlich sein könnte. Stattdessen ergibt sich erst durch die Wechselwirkung aller Teilprozesse ein Gesamtprozess, der auf einer ähnlichen Zeitskala wie die auftretende Flussantwort liegt. Einzige Kenngröße aller Teilprozesse ist dabei die Konzentration der Substrate in dem Kompartiment, sodass die Flussantwort wahrscheinlich konzentrationsgekoppelt ist. Es wurden verschiedene Sensoren für die Flussregulation getestet und festgestellt, dass der Sensor vermutlich die CO_2- oder HCO_3^--Konzentration im Plasma oder Interstitium ist. Zudem wurde eine Bayes'sche Datenanalyse mittels Metropolisalgorithmus in MatLab implementiert. Bei der Anwendung dieser auf die experimentell bestimmte Stoffumsatzrate konnte der Parameter v_{max} bestimmt werden und es zeigte sich, dass das Modell unzureichend ist, um die Abhängigkeit des Stoffumsatzes von der hyperkapnischen Reaktion abzubilden. Gründe dafür sind vermutlich die Annahmen von konstanten Parametern für Permeabilität, Volumina, Wasserlöslichkeit und/oder v_{max}. Da diese Annahmen jedoch in den meisten physiologisch bedingten mathematischen Modellen Anwendung finden [7], ist vielleicht sogar allgemein zu prüfen, ob diese Annahmen für dynamische Systeme anwendbar sind.

8 Ausblick

In Fortsetzung dieser Arbeit wäre denkbar, das Modell dahingehend zu adaptieren, dass auch andere Größen als die Konzentrationen und der Fluss als zeitabhängig aufgefasst werden. Dazu müssten zusätzliche Zusammenhänge, welche die Abhängigkeit dieser Größen von den Konzentrationen beschreiben, im Modell berücksichtigt werden. Bevor das Modell aber erweitert wird und dabei zusätzliche Parameter eingeführt werden, wäre es ratsam, zunächst weitere Parameter des aktuellen Modells mittels der implementierten Bayes'schen Datenanalyse zu bestimmen. Dabei ist zu prüfen, inwieweit das Modell für die Vorhersage anderer Größen als der Stoffumsatzrate, wie zum Beispiel die Ausflusskonzentrationen, überhaupt geeignet ist. Dafür sind weitere Experimente am isoliert perfundierten Mausherzen geplant. Zudem stehen auch PET-Daten zur Datenauswertung zur Verfügung. Bezüglich der Modellierung der biphasigen Flussantwort wäre eine Erweiterung des Modells um das Kompartiment der Endothelzellen und die Einführung von NO und der NO-Synthase denkbar. Für den ersten Peak erscheint weiterhin ein differentieller Regelkreis geeignet. Dessen Form könnte nach Sicherung der korrekten Parameterwerte durch die Bayes'sche Datenanalyse bezüglich des verwendeten Sensors noch genauer charakterisiert werden.

9 Zusammenfassung

In dieser Arbeit wurde ein orts- und zeitabhängigen Krogh-Zylinder-Modell, welches die Konzentrationen der Substanzen O_2, CO_2, HCO_3^-, H^+, $H_2PO_4^-$ und HPO_4^{2-} angewendet auf ein nach Langendorff isoliert perfundiertes Mausherz beschreibt, ein zeitabhängiges, jedoch ortsunabhängiges Stirred-Tank-Modell hergeleitet, das unter einmaliger Anwendung des Krogh-Zylinder-Modells jedoch trotzdem in der Lage ist, ortsabhängige Inhomogenitäten zu berücksichtigen. Es wurden mehrere mögliche Sensoren für die erste Phase der biphasigen Flussantwort, welche unter hyperkapnischen Bedingung auftritt, untersucht und CO_2 und HCO_3^- als Kandidaten der Ursache der Flussantwort ermittelt. Zudem wurde in MatLab eine Bayes'sche Datenanalyse mittels Metropolisalgorithmus implementiert, die dazu verwendet wurde, einige Modellparameter anhand experimenteller Daten zu ermitteln. Damit konnte ein Wert für v_{max} der Cytochrom-c-Oxidase unter physiologischen Bedingungen am Mausherzen bestimmt werden. Zudem wurden die Grenzen in den Voraussagen des Modells aufgezeigt, welche als Ausgangspunkt für zukünftige Erweiterungen des Modells dienen könnten.

10 Summary

This work is based on a model, which describes space- and time-distributed the concentrations of O_2, CO_2, HCO_3^-, H^+, $H_2PO_4^-$ und HPO_4^{2-} on an isolated perfused mouse heart in Langendorff configuration. Under hypercapnic conditions that mouse heart reacts with a biphasic change of the flow. In this work some possible sensors for the first phase of the flow change had been modelled and simulated with the result, that CO_2 and HCO_3^- are the most plausible candidates. Based on the sapce-distributed, so called Krogh-cylinder-model, a simpler, space independed Stirred-Tank-Model had been derived. This model needs to import information about the spaced-distibution from the Krogh-cylinder-Model but works independent after the onetime import of that information. In addition, a bayes analysis due to a Metropolis algorithm had been implemented in MatLab, which is used to predict the system parameter of the model based on concrete experimentel data. That way the parameter v_{max} of the Cytochrom-c-oxidase under physiologic conditions was performed. In the end, the limits of the predictions of the modell had been exposed, which sets the basis for the extention of the former model.

Anhang A: Quellcode

A.1 jSim-Modelle

A.1.1 Ausgangsmodell mit konstantem Fluss

Dieses Modell wurde in der Praktikumsarbeit [25] entwickelt und basiert auf einem von Dash und Bassingthwaighte [7] vorgestelltem Modell.

```
JSim v1.1

      import nsrunit;   unit conversion on;

      math o2co2model_v1 {

      realDomain t sec; // time domain
      t.min=0; t.max=300; t.delta=1;

      realDomain x cm;  // space domain
      real L=0.1 cm;
      int Ngrid=51;
      x.min=0; x.max=L; x.ct=Ngrid;
      private x.min, x.max, x.ct;

//-------------------------------------------------------------------
// Parameters for O2-CO2 transport, exchange, and metabolism
   real
      Fpl = 6.0 ml/(min*g),     // flow of plasma, ml/(min*g)
      Vpl = 0.07 ml/g,          // anatomical volume of plasma
      Visf = 0.20 ml/g,         // anatomical volume of ISF
      Vpc = 0.70 ml/g,          // anatomical volume of PCs
      Wpl = 0.94,               // fractional water content of plasma
      Wisf = 0.92,              // fractional water content of ISF
      Wpc = 0.80,               // fractional water content of PCs
      VWpl = Vpl*Wpl,           // volume of water content in plasma
      VWisf = Visf*Wisf,        // volume of water content in ISF
      VWpc = Vpc*Wpc,           // volume of water content in PCs
      RQ   = 0.80 dimensionless,// respiratory quotient (unitless)
      Km   = 7.0e-7 M,          // Km for O2 on cytochrome oxidase in PCs
      Vmax = 5.0e-6 mol/min/g,  // Vmax for O2 consumption by cytochrome oxidase in PCs

      O2PScap  = 200 ml/(min*g), // PS for O2 across CAP Membrane
      O2PSpc   = 2000 ml/(min*g),// PS for O2 across PC membrane
      CO2PScap = 200 ml/(min*g), // PS for CO2 across CAP Membrane
      CO2PSpc  = 2000 ml/(min*g),// PS for CO2 across PC membrane
      HpPScap  = 20 ml/(min*g),  // PS for H+ across CAP Membrane
      HpPSpc   = 200 ml/(min*g), // PS for H+ across PC Membrane
      HCO3mPScap = 20 ml/(min*g),// PS for HCO3- across CAP Membrane
      HCO3mPSpc = 200 ml/(min*g), // PS for HCO3- across PC membrane
        H2PO4mPScap  = 20 ml/(min*g),
        HPO42mPScap = 20 ml/(min*g),

      Rcap      = 0.63,          // Gibbs-Donnan ratio [H+]pl/[H+]isf
      Rpc       = 0.79,          // Gibbs-Donnan ratio [H+]isf/[H+]pc

      O2Dpl = 1e-4 cm^2/sec,    // diffusion coefficient for O2 in plasma
      O2Disf = 1e-4 cm^2/sec,   // diffusion coefficient for O2 in ISF
      O2Dpc = 1e-4 cm^2/sec,    // diffusion coefficient for O2 in PCs
      CO2Dpl = 1e-4 cm^2/sec,   // diffusion coefficient for CO2 in plasma
      CO2Disf = 1e-4 cm^2/sec,  // diffusion coefficient for CO2 in ISF
      CO2Dpc = 1e-4 cm^2/sec,   // diffusion coefficient for CO2 in PCs
      HpDpl = 1e-4 cm^2/sec,    // diffusion coefficient for H+ in plasma
      HpDisf = 1e-4 cm^2/sec,   // diffusion coefficient for H+ in ISF
      HpDpc = 1e-4 cm^2/sec,    // diffusion coefficient for H+ in PCs
      HCO3mDpl = 1e-4 cm^2/sec, // diffusion coefficient for HCO3- in plasma
      HCO3mDisf = 1e-4 cm^2/sec,// diffusion coefficient for HCO3- in ISF
      HCO3mDpc = 1e-4 cm^2/sec, // diffusion coefficient for HCO3- in PCs
        HPO42mDpl = 1e-4 cm^2/sec,
        HPO42mDisf = 1e-4 cm^2/sec,
        HPO42mDpc = 1e-4 cm^2/sec,
        H2PO4mDpl = 1e-4 cm^2/sec,
        H2PO4mDisf = 1e-4 cm^2/sec,
        H2PO4mDpc = 1e-4 cm^2/sec,

      BCpl = 6 mM,              // buffering capacity in plasma
```

```
      BCisf = 24 mM,              // buffering capacity in ISF
      BCpc = 45 mM,               // buffering capacity in PCs
      CFpl = 100,                 // catalytic factor in plasma due to CA
      CFisf = 5000,               // catalytic factor in ISF due to CA
      CFpc = 10000;               // catalytic factor in PCs due to CA
//-------------------------------------------------------------------------
// Rate and equilibrium constants for the reactions of O2 and CO2 with the
// myoglobin (privately fixed): used for computing saturations.
   real
      alphaO2 = 1.46e-6 M/mmHg, // O2 solubility coefficient
      alphaCO2 = 3.27e-5 M/mmHg,// CO2 solubility coefficient
      kp1     = 0.12 1/sec,     // forward rate constant in CO2+H2O reaction
      K1      = 4.47e-7 M,// EQUIL constant in overall CO2+H2O reaction
        kb1 = kp1/K1,
        kf2 = 0.12 1/sec,
        K2  = 6.2e-8 M,
        kb2 = kf2/K2,
      CMb     = 0.4e-3 M,         // CONC of Mb in PCs = 5/(16800*FCV) M
      CMbpc   = CMb/Wpc,          // CONC of Mb in the water space of PCs
      P50Mb   = 2.4 mmHg,         // P50 for MbO2 saturation
      NormConc = 1 M;             // H+ ion CONC normalization variable
//-------------------------------------------------------------------------
//jump function for the CO2 input into the system
extern real pCO2in(t) mmHg;
//-------------------------------------------------------------------------
// Physiological variables which are computed as solutions of the problem
   real
      pO2pl(t,x) mmHg,            // partial pressure of O2 in plasma
      pO2isf(t,x) mmHg,           // partial pressure of O2 in ISF
      pO2pc(t,x) mmHg,            // partial pressure of O2 in PCs
      pCO2pl(t,x) mmHg,           // partial pressure of CO2 in plasma
      pCO2isf(t,x) mmHg,          // partial pressure of CO2 in ISF
      pCO2pc(t,x) mmHg,           // partial pressure of CO2 in PCs
      O2Cpl(t,x) M,               // concentration of disolved O2 in plasma
      O2Cisf(t,x) M,              // concentration of disolved O2 in ISF
      O2Cpc(t,x) M,               // concentration of disolved O2 in PCs
      CO2Cpl(t,x) M,              // concentration of disolved CO2 in plasma
      CO2Cisf(t,x) M,             // concentration of disolved CO2 in ISF
      CO2Cpc(t,x) M,              // concentration of disolved CO2 in PCs

      KMbO2 M^(-1),               // Hill's coefficient for MbO2 saturation
      SMbO2(t,x),                 // MbO2 saturation (unitless)
      CMbO2(t,x) M,               // concentration of MbO2 in RBCs
      TO2Cpc(t,x) M,              // total O2 concentration in PCs
      O2Gpc(t,x) ml/min/g,        // gulosity for O2 consumption in PCS
      MRO2pc(t,x) mol/min/g,      // O2 consumption rate in PCS
      MRO2pcavg(t) mol/min/g,     // AVG O2 consumption rate in PCS
      O2Cart(t) M,                // O2 concentration in arterial blood
      O2Cven(t) M,                // O2 concentration in venous blood
      O2extr(t),                  // O2 extraction (percentage; unitless)

      HCO3mCpl(t,x) M,            // concentration of HCO3- in plasma
      HCO3mCisf(t,x) M,           // concentration of HCO3- in ISF
      HCO3mCpc(t,x) M,            // concentration of HCO3- in PCs
      HpCpl(t,x) M,               // concentration of H+ in plasma
      HpCisf(t,x) M,              // concentration of H+ in ISF
      HpCpc(t,x) M,               // concentration of H+ in PCs
      pHpl(t,x),                  // pH in plasma (unitless)
      pHisf(t,x),                 // pH in ISF (unitless)
      pHpc(t,x),                  // pH in PCs (unitless)

        H2PO4mCpl(t,x) M,
        H2PO4mCisf(t,x) M,
        H2PO4mCpc(t,x) M,
        HPO42mCpl(t,x) M,
        HPO42mCisf(t,x) M,
        HPO42mCpc(t,x) M;

//-------------------------------------------------------------------------
// Physiological Variables as mean value
real
        apO2pl(t) mmHg,           // partial pressure of O2 in plasma
      apO2isf(t) mmHg,            // partial pressure of O2 in ISF
      apO2pc(t) mmHg,             // partial pressure of O2 in PCs
      apCO2pl(t) mmHg,            // partial pressure of CO2 in plasma
      apCO2isf(t) mmHg,           // partial pressure of CO2 in ISF
      apCO2pc(t) mmHg,            // partial pressure of CO2 in PCs

      aHCO3mCpl(t) M,             // concentration of HCO3- in plasma
      aHCO3mCisf(t) M,            // concentration of HCO3- in ISF
      aHCO3mCpc(t) M,             // concentration of HCO3- in PCs
      apHpl(t),                   // pH in plasma (unitless)
      apHisf(t),                  // pH in ISF (unitless)
      apHpc(t),                   // pH in PCs (unitless)
        aHpCpl(t) M,
        aHpCisf(t) M,
        aHpCpc(t) M;
```

```
//calculations of mean
      apO2pl = 1/L*integral(pO2pl@x);
      apO2isf = 1/L*integral(pO2isf@x);
      apO2pc= 1/L*integral(pO2pc@x);
      apCO2pl= 1/L*integral(pCO2pl@x);
      apCO2isf= 1/L*integral(pCO2isf@x);
      apCO2pc= 1/L*integral(pCO2pc@x);

      aHCO3mCpl= 1/L*integral(HCO3mCpl@x);
      aHCO3mCisf= 1/L*integral(HCO3mCisf@x);
      aHCO3mCpc= 1/L*integral(HCO3mCpc@x);
      apHpl= 1/L*integral(pHpl@x);
      apHisf= 1/L*integral(pHisf@x);
      apHpc= 1/L*integral(pHpc@x);

        aHpCpl = 10^(-apHpl)*NormConc;
        aHpCisf = 10^(-apHisf)*NormConc;
        aHpCpc = 10^(-apHpc)*NormConc;

real
        aH2PO4mCpl(t) = 1/L*integral(H2PO4mCpl@x),
        aHPO42mCpl(t) = 1/L*integral(HPO42mCpl@x);
//-------------------------------------------------------------------------------
// Calculations for evaluating the initial (t=t.min=0) conditions
   real          // variables in the BTEX unit at time t=0
      pO2plt0   = 100 mmHg,               // pO2 in plasma at t=0
      pHplt0    = 7.4,                    // pH in plasma at t=0
      pHisft0   = pHplt0 + log(Rcap),     // pH in ISF at t=0
      pHpct0    = pHisft0 + log(Rpc),     // pH in PCs at t=0
        H2PO4mCplt0 = 1e-3 M,
        H2PO4mCpct0 = 60e-3 M;

    real                  // CALCULATIONS AT TIME t=0.
        tO2Cpl0 M,        tO2Cisf0 M,       tO2Cpc0 M,
        tCO2Cpl0 M,       tCO2Cisf0 M,     tCO2Cpc0 M,
        tO2CplL M,        tO2CisfL M,       tO2CpcL M,
        tCO2CplL M,       tCO2CisfL M,     tCO2CpcL M,
        HpCplt0 M,        HpCisft0 M,      HpCpct0 M,
        HCO3mCplt0 M,     HCO3mCisft0 M,  HCO3mCpct0 M,
        H2PO4mCisft0 M,  H2PO4mCpct0 M,
        HPO42mCplt0 M,  HPO42mCisft0 M,  HPO42mCpct0 M,
        KMbO2t0 M^(-1),
        tSMbO20,
        tSMbO2L,
        tCMbO20 M,
        tCMbO2L M,
        tTO2Cpc0 M,
        tTO2CpcL M,
      tO2Cpl0    = alphaO2*pO2plt0;
      tO2Cisf0   = tO2Cpl0-Vmax/O2PScap;
      tO2Cpc0    = tO2Cpl0-Vmax*(1/O2PScap+1/O2PSpc);
      tCO2Cpl0   = alphaCO2*pCO2in(t.min);
        tCO2Cisf0=tCO2Cpl0+Vmax*RQ/CO2PScap;
        tCO2Cpc0=tCO2Cisf0+Vmax*RQ/CO2PSpc;

        tO2CplL = tO2Cpl0-Vmax/(Wpl*Fpl);
        tO2CisfL = tO2Cisf0-Vmax/(Wpl*Fpl);
        tO2CpcL = tO2Cpc0-Vmax/(Wpl*Fpl);
        tCO2CplL = tCO2Cpl0 + RQ*Vmax/(Wpl*Fpl);
        tCO2CisfL = tCO2Cisf0 + RQ*Vmax/(Wpl*Fpl);
        tCO2CpcL = tCO2Cpc0 + RQ*Vmax/(Wpl*Fpl);

      HpCplt0    = NormConc*10^(-pHplt0);
      HpCisft0   = NormConc*10^(-pHisft0);
      HpCpct0    = NormConc*10^(-pHpct0);
      HCO3mCplt0 = K1*tCO2Cpl0/HpCplt0;
      HCO3mCisft0 = K1*tCO2Cisf0/HpCisft0;
      HCO3mCpct0 = K1*tCO2Cpc0/HpCpct0;
      KMbO2t0   = (alphaO2*P50Mb)^(-1);
        tSMbO20   = KMbO2t0*tO2Cpc0/(1+KMbO2t0*tO2Cpc0);
        tSMbO2L   = KMbO2t0*tO2CpcL/(1+KMbO2t0*tO2CpcL);
        tCMbO20   = CMbpc*tSMbO20;
        tCMbO2L   = CMbpc*tSMbO2L;
        tTO2Cpc0    = tO2Cpc0+tCMbO20;
        tTO2CpcL    = tO2CpcL+tCMbO2L;

        H2PO4mCisft0 = H2PO4mCplt0/Rcap;

        HPO42mCplt0 = K2*H2PO4mCplt0/HpCplt0;
        HPO42mCisft0 = K2*H2PO4mCisft0/HpCisft0;
        HPO42mCpct0 = K2*H2PO4mCpct0/HpCpct0;

//-------------------------------------------------------------------------------
// Initial (t=t.min=0) and boundary (x=x.min=0 & x=x.max=L) conditions. The
// above derived boundary conditions are used at the arterial end (x=0) of
// the capillary. Zero flux conditions are used at the venous end (x=L) of
// the capillary as well as at x=0 and x=L of the tissue region.
   when (t=t.min) {      // INITIAL CONDITIONS
```

```
    O2Cpl = if(x = x.min) tO2Cpl0 else tO2Cpl0*(L-x)/L + tO2CplL*x/L;
    O2Cisf = tO2Cisf0*(L-x)/L + tO2CisfL*x/L;
    TO2Cpc = tTO2Cpc0*(L-x)/L + tTO2CpcL*x/L;
    CO2Cpl = if(x = x.min) alphaCO2*pCO2in else tCO2Cpl0*(L-x)/L + tCO2CplL*x/L;
    CO2Cisf = tCO2Cisf0*(L-x)/L + tCO2CisfL*x/L;
    CO2Cpc = tCO2Cpc0*(L-x)/L + tCO2CpcL*x/L;
    HCO3mCpl = HCO3mCplt0;
    HCO3mCisf = HCO3mCisft0;
    HCO3mCpc = HCO3mCpct0;
    HpCpl = HpCplt0;
    HpCisf = HpCisft0;
    HpCpc = HpCpct0;
      HPO42mCpl = HPO42mCplt0;
      HPO42mCisf = HPO42mCisft0;
      HPO42mCpc = HPO42mCpct0;
      H2PO4mCpl = H2PO4mCplt0;
      H2PO4mCisf = H2PO4mCisft0;
      H2PO4mCpc = H2PO4mCpct0;
  }

  when (x=x.min) {      // LEFT BOUNDARY CONDITIONS
    (-Fpl*L/Vpl)*(O2Cpl-tO2Cpl0)+O2Dpl*O2Cpl:x =0;
    O2Disf*O2Cisf:x =0;
    O2Dpc*TO2Cpc:x =0;
    (-Fpl*L/Vpl)*(CO2Cpl-alphaCO2*pCO2in)+CO2Dpl*CO2Cpl:x = 0;
    CO2Disf*CO2Cisf:x =0;
    CO2Dpc*CO2Cpc:x =0;
    (-Fpl*L/Vpl)*(HCO3mCpl-HCO3mCplt0)+HCO3mDpl*HCO3mCpl:x =0;
    HCO3mDisf*HCO3mCisf:x =0;
    HCO3mDpc*HCO3mCpc:x =0;
    (-Fpl*L/Vpl)*(HpCpl-HpCplt0)+HpDpl*HpCpl:x =0;
    HpDisf*HpCisf:x =0;
    HpDpc*HpCpc:x =0;
      (-Fpl*L/Vpl)*(HPO42mCpl-HPO42mCplt0)+HPO42mDpl*HPO42mCpl:x =0;
      HPO42mDisf*HPO42mCisf:x = 0;
      HPO42mDpc*HPO42mCpc:x = 0;
      (-Fpl*L/Vpl)*(H2PO4mCpl-H2PO4mCplt0)+H2PO4mDpl*H2PO4mCpl:x =0;
      H2PO4mDisf*H2PO4mCisf:x = 0;
      H2PO4mDpc*H2PO4mCpc:x = 0;

    O2Cart = Wpl*O2Cpl;
  }

  when (x=x.max) {      // RIGHT BOUNDARY CONDITIONS
    O2Dpl*O2Cpl:x    = 0;
    O2Cisf:x         = 0;
    TO2Cpc:x         = 0;
    CO2Dpl*CO2Cpl:x  = 0;
    CO2Cisf:x        = 0;
    CO2Cpc:x         = 0;
    HCO3mDpl*HCO3mCpl:x = 0;
    HCO3mCisf:x      = 0;
    HCO3mCpc:x       = 0;
    HpDpl*HpCpl:x    = 0;
    HpCisf:x         = 0;
    HpCpc:x          = 0;
      H2PO4mDpl*H2PO4mCpl:x = 0;
      H2PO4mCisf:x = 0;
      H2PO4mCpc:x = 0;
      HPO42mDpl*HPO42mCpl:x = 0;
      HPO42mCisf:x = 0;
      HPO42mCpc:x = 0;

    O2Cven = Wpl*O2Cpl;
    O2extr = (O2Cart - O2Cven)/O2Cart;
    MRO2pcavg = Fpl*O2extr*O2Cart;
  }

//-----------------------------------------------------------------------
// Equations for Hill coefficient KMbO2, saturation SMbO2, and concentration CMbO2. KMbO2 is the
// equilibrium constant in Mb+O2 <--> MbO2 reaction.
    KMbO2  = (alphaO2*P50Mb)^(-1);
    SMbO2  = KMbO2*O2Cpc/(1+KMbO2*O2Cpc);
    CMbO2  = CMbpc*SMbO2;

//-----------------------------------------------------------------------
// Compute the partial pressures of O2 and CO2 and pH in the three regions
// of the BTEX unit
    pO2pl  = O2Cpl/alphaO2;
    pO2isf = O2Cisf/alphaO2;
    pO2pc  = O2Cpc/alphaO2;
    pCO2pl  = CO2Cpl/alphaCO2;
    pCO2isf = CO2Cisf/alphaCO2;
    pCO2pc  = CO2Cpc/alphaCO2;
    pHpl  = -log(HpCpl/NormConc);
    pHisf = -log(HpCisf/NormConc);
    pHpc  = -log(HpCpc/NormConc);
//-----------------------------------------------------------------------
```

```
// PDEs for O2 transport in plasma, ISF, and parenchymal cells, where
// the relation in parenchymal cells [i.e., TO2Cpc = O2Cpc + CMbpc*SMbO2(O2Cpc)] leads to
// a quadratic equation for O2Cpc. It is solved analytically to find O2Cpc.

      O2Cpl:t = - (Fpl/Vpl)*L*(O2Cpl:x)
                - (O2PScap/VWpl)*(O2Cpl-O2Cisf)
                + O2Dpl*(O2Cpl:x:x);

      O2Cisf:t =   (O2PScap/VWisf)*(O2Cpl-O2Cisf)
                 - (O2PSpc/VWisf)*(O2Cisf-O2Cpc)
                 + O2Disf*(O2Cisf:x:x);

      TO2Cpc:t =   (O2PSpc/VWpc)*(O2Cisf-O2Cpc)
                 + O2Dpc*(TO2Cpc:x:x) - (O2Gpc/VWpc)*O2Cpc;

      O2Cpc = if (TO2Cpc <= 0) 0
              else (((1+KMbO2*(CMbpc-TO2Cpc))^2 + 4*KMbO2*TO2Cpc)^0.5
                   - (1+KMbO2*(CMbpc-TO2Cpc)))/(2*KMbO2);

      O2Gpc = Vmax/(Km+O2Cpc);
      MRO2pc = O2Gpc*O2Cpc;

//-------------------------------------------------------------------------
// PDEs for CO2 in plasma, ISF, and parenchymal cells.

      CO2Cpl:t = - (Fpl/Vpl)*L*(CO2Cpl:x)
                 - (CO2PScap/VWpl)*(CO2Cpl-CO2Cisf)
                 + CO2Dpl*(CO2Cpl:x:x)
                 - CFpl*(kp1*CO2Cpl-kb1*HCO3mCpl*HpCpl);

      CO2Cisf:t =   (CO2PScap/VWisf)*(CO2Cpl-CO2Cisf)
                  - (CO2PSpc/VWisf)*(CO2Cisf-CO2Cpc)
                  + CO2Disf*(CO2Cisf:x:x)
                  - CFisf*(kp1*CO2Cisf-kb1*HCO3mCisf*HpCisf);

      CO2Cpc:t =   (CO2PSpc/VWpc)*(CO2Cisf-CO2Cpc)
                 + CO2Dpc*(CO2Cpc:x:x) + RQ*(O2Gpc/VWpc)*O2Cpc
                 - CFpc*(kp1*CO2Cpc-kb1*HCO3mCpc*HpCpc);

//-------------------------------------------------------------------------
// PDEs for HCO3- ions in plasma, ISF, and parenchymal cells.

      HCO3mCpl:t = - (Fpl/Vpl)*L*(HCO3mCpl:x)
                   - (HCO3mPScap/VWpl)*(Rcap*HCO3mCpl-HCO3mCisf)
                   + HCO3mDpl*(HCO3mCpl:x:x)
                   + CFpl*(kp1*CO2Cpl-kb1*HCO3mCpl*HpCpl);

      HCO3mCisf:t =   (HCO3mPScap/VWisf)*(Rcap*HCO3mCpl-HCO3mCisf)
                    - (HCO3mPSpc/VWisf)*(Rpc*HCO3mCisf-HCO3mCpc)
                    + HCO3mDisf*(HCO3mCisf:x:x)
                    + CFisf*(kp1*CO2Cisf-kb1*HCO3mCisf*HpCisf);

      HCO3mCpc:t =   (HCO3mPSpc/VWpc)*(Rpc*HCO3mCisf-HCO3mCpc)
                   + HCO3mDpc*(HCO3mCpc:x:x)
                   + CFpc*(kp1*CO2Cpc-kb1*HCO3mCpc*HpCpc);

//-------------------------------------------------------------------------
// PDEs for HPO42- ions in plasma, ISF, and parenchymal cells.

      HPO42mCpl:t = - (Fpl/Vpl)*L*(HPO42mCpl:x)
                    - (HPO42mPScap/VWpl)*(Rcap^2*HPO42mCpl-HPO42mCisf)
                    + HPO42mDpl*(HPO42mCpl:x:x)
                    + (kf2*H2PO4mCpl-kb2*HPO42mCpl*HpCpl);

      HPO42mCisf:t =   (HPO42mPScap/VWisf)*(Rcap^2*HPO42mCpl-HPO42mCisf)
                     //- (HPO42mPSpc/VWisf)*(Rpc^2*HPO42mCisf-HPO42mCpc)
                     + HPO42mDisf*(HPO42mCisf:x:x)
                     +(kf2*H2PO4mCisf-kb2*HPO42mCisf*HpCisf);

      HPO42mCpc:t =   //(HPO42mPSpc/VWpc)*(Rpc^2*HPO42mCisf-HPO42mCpc)+
                        HPO42mDpc*(HPO42mCpc:x:x)
                    + (kf2*H2PO4mCpc-kb2*HPO42mCpc*HpCpc);

//-------------------------------------------------------------------------
// PDEs for H2PO4- ions in plasma, ISF, and parenchymal cells.

      H2PO4mCpl:t = - (Fpl/Vpl)*L*(H2PO4mCpl:x)
                    - (H2PO4mPScap/VWpl)*(Rcap*H2PO4mCpl-H2PO4mCisf)
                    + H2PO4mDpl*(H2PO4mCpl:x:x)
                    - (kf2*H2PO4mCpl-kb2*HPO42mCpl*HpCpl);

      H2PO4mCisf:t =   (H2PO4mPScap/VWisf)*(Rcap*H2PO4mCpl-H2PO4mCisf)
                     //- H2PO4mPSpc/VWisf*(Rpc*H2PO4mCisf-H2PO4mCpc)
                     + H2PO4mDisf*(H2PO4mCisf:x:x)
                     - (kf2*H2PO4mCisf-kb2*HPO42mCisf*HpCisf);

      H2PO4mCpc:t =   //(H2PO4mPSpc/VWpc)*(Rpc*H2PO4mCisf-H2PO4mCpc)+
                        H2PO4mDpc*(H2PO4mCpc:x:x)
```

```
                    - (kf2*H2PO4mCpc-kb2*HPO42mCpc*HpCpc);

//-------------------------------------------------------------------------
// PDEs for H+ ions in plasma, ISF, and parenchymal cells.

      HpCpl:t  = - (Fpl/Vpl)*L*(HpCpl:x)
                 - (HpPScap/VWpl)*(HpCpl-Rcap*HpCisf)
                 + HpDpl*(HpCpl:x:x) + (2.303/BCpl)*HpCpl
                 * CFpl*(kp1*CO2Cpl-kb1*HCO3mCpl*HpCpl)
                + (2.303/BCpl)*HpCpl
                 * (kf2*H2PO4mCpl-kb2*HPO42mCpl*HpCpl);

      HpCisf:t =   (HpPScap/VWisf)*(HpCpl-Rcap*HpCisf)
                 - (HpPSpc/VWisf)*(HpCisf-Rpc*HpCpc)
                 + HpDisf*(HpCisf:x:x) + (2.303/BCisf)*HpCisf
                 * CFisf*(kp1*CO2Cisf-kb1*HCO3mCisf*HpCisf)
                + (2.303/BCisf)*HpCisf
                 * (kf2*H2PO4mCisf-kb2*HPO42mCisf*HpCisf);

      HpCpc:t =   (HpPSpc/VWpc)*(HpCisf-Rpc*HpCpc)
                 + HpDpc*(HpCpc:x:x) + (2.303/BCpc)*HpCpc
                 * CFpc*(kp1*CO2Cpc-kb1*HCO3mCpc*HpCpc)
                + (2.303/BCpc)*HpCpc
                 * (kf2*H2PO4mCpc-kb2*HPO42mCpc*HpCpc);

//-------------------------------------------------------------------------
//Calculations of the time dependent transposition factors for the stirred tank modell

real fO2(t) = (pO2plt0-pO2pl(t,L))/(pO2plt0-apO2pl);
real fCO2(t) = (pCO2in-pCO2pl(t,L))/(pCO2in-apCO2pl);
real fHCO3m(t) = (HCO3mCplt0-HCO3mCpl(t,L))/(HCO3mCplt0-aHCO3mCpl);
real fH2PO4m(t) = (H2PO4mCplt0-H2PO4mCpl(t,L))/(H2PO4mCplt0-aH2PO4mCpl);
real fHPO42m(t) = (HPO42mCplt0-HPO42mCpl(t,L))/(HPO42mCplt0-aHPO42mCpl);
real fHp(t) = (HpCplt0-HpCpl(t,L))/(HpCplt0-aHpCpl);
}
//-------------------------------------------------------------------------

//-------------------------------------------------------------------------
/*
  DETAILED DESCRIPTION:
  A detailed nonlinear three-region (plasma, interstitial fluid, and
  parenchymal cell) axially distributed convection-diffusion-permeation-reaction-binding
  computational model is developed to study the simultaneous transport and exchange of
  oxygen (O2) and carbon dioxide (CO2) in the blood-tissue exchange system of an organ.
  Since the pH variation in blood and tissue influences the transport and exchange of O2
  and CO2 (Bohr and Haldane effects), and since most CO2 is transported as HCO3- (bicarbonate)
  via the CO2 hydration (buffering) reaction, the transport and exchange of HCO3- and H+ are
  also simulated along with that of O2 and CO2. Furthermore, the model accounts for  myoglobin-facilitated transport
  of O2 inside the parenchymal cells. The consumption of O2 through cytochrome-C oxidase reaction
  inside the parenchymal cells is based on Michaelis-Menten kinetics. The corresponding
  production of CO2 is determined by respiratory quotient (RQ), depending on the relative
  consumption of carbohydrate, protein, and fat. The model gives a physiologically realistic
  description of O2 transport and metabolism in the microcirculation of the organ.

  This Model is based on the following Modell:
    Dash, R.K. and Bassingwaighte, J.B., Simultaneous Blood-Tissue Exchange of Oxygen, Carbon Dioxide,
    Bicarbonate, and Hydrogen Ion, Ann Biomed Eng 34(7): 1129-1148 2006.

 COPYRIGHT AND REQUEST FOR ACKNOWLEDGMENT OF USE:
  Copyright (C) 1999-2009 University of Washington. From the National Simulation Resource,
  Director J. B. Bassingthwaighte, Department of Bioengineering, University of Washington, Seattle WA 98195-5061.
  Academic use is unrestricted. Software may be copied so long as this copyright notice is included.

  This software was developed with support from NIH grant HL073598.
  Please cite this grant in any publication for which this software is used and if possible
  send one reprint to the address given above OR send an email giving the full reference to jbb2@uw.edu..
*/
```

A.1.2 Modelladaptionen für diffusiven Einfluss und Flussregulation

```
%Als Ersatz f"ur Fpl Deklaration
Fpl0 =6.0 ml/(min*g),    // flow of plasma, ml/(min*g)

%"Anderung der Left-Boundary-Condition f"ur CO2Cpl
when (x=x.min) {         // LEFT BOUNDARY CONDITIONS
      (-Fpl*L/Vpl)*(CO2Cpl-alphaCO2*FF)+CO2Dpl*CO2Cpl:x = 0;


%Im Quellcode zu erg"anzen
//exmpiric flow response with diffusion in the input signal
real gewicht = 5 g;

extern real Cin(t) mmHg;
real Vein = 10 ml;
```

```
real F = Fpl0*gewicht;
real tmean = Vein/F;
real RD = 0.2;
real offset = 40 mmHg;
real RDp = 0.48;
real tp = RD/RDp*tmean;
real td = tmean*(1-RD/RDp);
real C1(t) mmHg;
C1(t) = if(t>=tmean) Cin(t-tmean) else offset;
real w1 = 2*Pi/tp;
real xsi1 = 0.95;
real w2 = 1.82*w1;
real xsi2 = 0.8;
real SD = RDp*tp;
real C2(t) mmHg;
real FF(t) mmHg;

when(t=t.min){
        C2 = offset;
        C2:t = 0;
        FF = offset;
        FF:t = 0;
}

C2:t:t + 2*xsi1*w1*C2:t+w1^2*C2 = w1^2*C1;
FF:t:t + 2*xsi2*w2*FF:t+w2^2*FF = w2^2*C2;


real regCO2pl = 1; //Controlling of which sensor to use for the first flow response
real regCO2isf = 0;
real regHCO3mpl = 0;
real regHCO3misf = 0;
real    factor1 = 7000 s, //magnification of the regulating system
        factor2 = 9000 s,
        factor3 = 2000 s,
        factor4 = 2200 s;
real d(t) = if(t<t.min+2*t.delta) 1 else if(t<0) 1
else (1+regCO2pl*abs((aCO2Cpl(t-t.delta)-aCO2Cpl(t-2*t.delta))/t.delta/NormConc)*factor1
+regCO2isf*abs((aCO2Cisf(t-t.delta)-aCO2Cisf(t-2*t.delta))/t.delta/NormConc)*factor2
+regHCO3mpl*abs((aHCO3mCpl(t-t.delta)-aHCO3mCpl(t-2*t.delta))/t.delta/NormConc)*factor3
+regHCO3misf*abs((aHCO3mCisf(t-t.delta)-aHCO3mCisf(t-2*t.delta))/t.delta/NormConc)*factor4);
real Fpl(t) = Fpl0*d;
```

A.2 Python-Modell mit Fipy

```
from fipy import Grid1D, CellVariable, TransientTerm, DiffusionTerm, ConvectionTerm, ImplicitSourceTerm, Viewer

#Loesungsgitter
dt = 1.e-3
t = 0
tmax = 5

m = Grid1D(nx=50, Lx=.1)

#Parameterdefinition
l = 0.1
f = 6.0/(60*1000)
vpl = 0.07/1000
wpl = 0.94
vwpl = vpl*wpl
vwisf = 0.2*0.92/1000
vwpc = 0.7*0.8/1000

rq = 0.8
km = 7e-7
vmax = 2.5e-6/60

o2pscap = 200.0/(60*1000)
o2pspc = 2000.0/(60*1000)
co2pscap = 200.0/(60*1000)
co2pspc = 2000.0/(60*1000)
hppscap = 20.0/(60*1000)
hppspc = 200.0/(60*1000)
hco3mpscap = 20.0/(60*1000)
hco3mpspc = 200.0/(60*1000)

rcap = 0.63
rpc = 0.79

o2dpl = 1e-4
o2disf = 1e-4
o2dpc = 1e-4
co2dpl = 1e-4
co2disf = 1e-4
co2dpc = 1e-4
```

```
hpdpl = 1e-4
hpdisf = 1e-4
hpdpc = 1e-4
hco3mdpl = 1e-4
hco3mdisf = 1e-4
hco3mdpc = 1e-4

betapl = 6.0/1000
betaisf = 24.0/1000
betapc = 45.0/1000

cfpl = 100
cfisf = 5000
cfpc = 10000

alphaO2 = 1.46e-6
alphaCO2 = 3.27e-5

kf1 = 0.12
k1 = 4.47e-7
kb1 = kf1/k1

fluss = (f*l/vpl,)

#Funktionsdefiniton mit Startwert
o2cpl = CellVariable(mesh=m, hasOld=True, value=634*alphaO2)
o2cisf = CellVariable(mesh=m, hasOld=True, value=634*alphaO2)
o2cpc = CellVariable(mesh=m, hasOld=True, value=634*alphaO2)

co2cpl = CellVariable(mesh=m, hasOld=True, value=40*alphaCO2)
co2cisf = CellVariable(mesh=m, hasOld=True, value=40*alphaCO2)
co2cpc = CellVariable(mesh=m, hasOld=True, value=40*alphaCO2)

hpcpl = CellVariable(mesh=m, hasOld=True, value=10**(-7.4))
hpcisf = CellVariable(mesh=m, hasOld=True, value=10**(-7.4)*rcap)
hpcpc = CellVariable(mesh=m, hasOld=True, value=10**(-7.4)*rcap*rpc)

hco3mcpl = CellVariable(mesh=m, hasOld=True, value=k1*40*alphaCO2/(10**(-7.4)))
hco3mcisf = CellVariable(mesh=m, hasOld=True, value=k1*40*alphaCO2/(10**(-7.4)*rcap))
hco3mcpc = CellVariable(mesh=m, hasOld=True, value=k1*40*alphaCO2/(10**(-7.4)*rcap*rpc))

#Festlegung der Ergebnisdiagramme
viO2 = Viewer((o2cpl, o2cisf, o2cpc))
viCO2 = Viewer((co2cpl, co2cisf, co2cpc))
viHp = Viewer((hpcpl, hpcisf, hpcpc))
viHCO3m = Viewer((hco3mcpl, hco3mcisf, hco3mcpc))

#Festlegung der Randbedingungen
o2cpl.constrain(634*alphaO2, m.facesLeft)
o2cpl.faceGrad.constrain(0, m.facesRight)
o2cisf.faceGrad.constrain(0, m.facesLeft)
o2cisf.faceGrad.constrain(0, m.facesRight)
o2cpc.faceGrad.constrain(0, m.facesLeft)
o2cpc.faceGrad.constrain(0, m.facesRight)

co2cpl.constrain(40*alphaCO2, m.facesLeft)
co2cpl.faceGrad.constrain(0, m.facesRight)
co2cisf.faceGrad.constrain(0, m.facesLeft)
co2cisf.faceGrad.constrain(0, m.facesRight)
co2cpc.faceGrad.constrain(0, m.facesLeft)
co2cpc.faceGrad.constrain(0, m.facesRight)

hpcpl.constrain(10**(-7.4), m.facesLeft)
hpcpl.faceGrad.constrain(0, m.facesRight)
hpcisf.faceGrad.constrain(0, m.facesLeft)
hpcisf.faceGrad.constrain(0, m.facesRight)
hpcpc.faceGrad.constrain(0, m.facesLeft)
hpcpc.faceGrad.constrain(0, m.facesRight)

hco3mcpl.constrain(k1*40*alphaCO2/(10**(-7.4)), m.facesLeft)
hco3mcpl.faceGrad.constrain(0, m.facesRight)
hco3mcisf.faceGrad.constrain(0, m.facesLeft)
hco3mcisf.faceGrad.constrain(0, m.facesRight)
hco3mcpc.faceGrad.constrain(0, m.facesLeft)
hco3mcpc.faceGrad.constrain(0, m.facesRight)

#Definition der Differentialgleichungen
o2pl = (TransientTerm(var=o2cpl) == DiffusionTerm(o2dpl,var=o2cpl) - ConvectionTerm(fluss, var=o2cpl)
-ImplicitSourceTerm(o2pscap/vwpl,var=o2cpl) + ImplicitSourceTerm(o2pscap/vwpl, var=o2cisf))
o2isf = (TransientTerm(var=o2cisf) == DiffusionTerm(o2disf, var=o2cisf) +
ImplicitSourceTerm(o2pscap/vwisf, var=o2cpl) - ImplicitSourceTerm(o2pscap/vwisf+o2pspc/vwisf, var=o2cisf)
+ ImplicitSourceTerm(o2pspc/vwisf, var=o2cpc))
o2pc = (TransientTerm(var=o2cpc) == DiffusionTerm(o2dpc, var= o2cpc)
+ ImplicitSourceTerm(o2pspc/vwpc, var=o2cisf) - ImplicitSourceTerm(o2pspc/vwpc, var=o2cpc)
- ImplicitSourceTerm(vmax/vwpc*1/(o2cpc+km), var=o2cpc))

co2pl = (TransientTerm(var=co2cpl) == DiffusionTerm(co2dpl,var=co2cpl)
- ConvectionTerm(fluss, var=co2cpl)
```

```
-ImplicitSourceTerm(co2pscap/vwpl,var=co2cpl) + ImplicitSourceTerm(co2pscap/vwpl, var=co2cisf)
- ImplicitSourceTerm(cfpl*kf1, var=co2cpl) + ImplicitSourceTerm(cfpl*kb1*hco3mcpl, var=hpcpl))
co2isf = (TransientTerm(var=co2cisf) == DiffusionTerm(co2disf, var=co2cisf)
+ ImplicitSourceTerm(co2pscap/vwisf, var=co2cpl) - ImplicitSourceTerm(co2pscap/vwisf+co2pspc/vwisf, var=co2cisf)
+ ImplicitSourceTerm(co2pspc/vwisf, var=co2cpc)
- ImplicitSourceTerm(cfisf*kf1, var=co2cisf) + ImplicitSourceTerm(cfisf*kb1*hco3mcisf, var=hpcisf))
co2pc = (TransientTerm(var=co2cpc) == DiffusionTerm(co2dpc, var= co2cpc)
+ ImplicitSourceTerm(co2pspc/vwpc, var=co2cisf) - ImplicitSourceTerm(co2pspc/vwpc, var=co2cpc)
+ ImplicitSourceTerm(rq*vmax/vwpc*1/(o2cpc+km), var=o2cpc) - ImplicitSourceTerm(cfpc*kf1, var=co2cpc)
+ ImplicitSourceTerm(cfpc*kb1*hco3mcpc, var=hpcpc))

hppl = (TransientTerm(var=hpcpl) == DiffusionTerm(hpdpl,var=hpcpl) - ConvectionTerm(fluss, var=hpcpl)
-ImplicitSourceTerm(hppscap/vwpl,var=hpcpl) + ImplicitSourceTerm(rcap*hppscap/vwpl, var=hpcisf)
+ ImplicitSourceTerm(2.303/betapl*hpcpl*cfpl*kf1, var=co2cpl)
- ImplicitSourceTerm(2.303/betapl*hpcpl*cfpl*kb1*hco3mcpl, var=hpcpl))
hpisf = (TransientTerm(var=hpcisf) == DiffusionTerm(hpdisf, var=hpcisf) + ImplicitSourceTerm(hppscap/vwisf, var=hpcpl)
- ImplicitSourceTerm(rcap*hppscap/vwisf+hppspc/vwisf, var=hpcisf) + ImplicitSourceTerm(rpc*hppspc/vwisf, var=hpcpc)
+ ImplicitSourceTerm(2.303/betaisf*hpcisf*cfisf*kf1, var=co2cisf)
- ImplicitSourceTerm(2.303/betaisf*hpcisf*cfisf*kb1*hco3mcisf, var=hpcisf))
hppc = (TransientTerm(var=hpcpc) == DiffusionTerm(hpdpc, var= hpcpc) + ImplicitSourceTerm(hppspc/vwpc, var=hpcisf)
- ImplicitSourceTerm(rpc*hppspc/vwpc, var=hpcpc) + ImplicitSourceTerm(2.303/betapc*hpcpc*cfpc*kf1, var=co2cpc)
- ImplicitSourceTerm(2.303/betapc*hpcpc*cfpc*kb1*hco3mcpc, var=hpcpc))

hco3mpl = (TransientTerm(var=hco3mcpl) == DiffusionTerm(hco3mdpl,var=hco3mcpl)
- ConvectionTerm(fluss, var=hco3mcpl)
-ImplicitSourceTerm(rcap*hco3mpscap/vwpl,var=hco3mcpl) + ImplicitSourceTerm(hco3mpscap/vwpl, var=hco3mcisf)
+ ImplicitSourceTerm(cfpl*kf1, var=co2cpl) - ImplicitSourceTerm(cfpl*kb1*hco3mcpl, var=hpcpl))
hco3misf = (TransientTerm(var=hco3mcisf) == DiffusionTerm(hco3mdisf, var=hco3mcisf)
+ ImplicitSourceTerm(rcap*hco3mpscap/vwisf, var=hco3mcpl)
- ImplicitSourceTerm(hco3mpscap/vwisf+rpc*hco3mpspc/vwisf, var=hco3mcisf)
+ ImplicitSourceTerm(hco3mpspc/vwisf, var=hco3mcpc) + ImplicitSourceTerm(cfisf*kf1, var=co2cisf)
- ImplicitSourceTerm(cfisf*kb1*hco3mcisf, var=hpcisf))
hco3mpc = (TransientTerm(var=hco3mcpc) == DiffusionTerm(hco3mdpc, var= hco3mcpc)
+ ImplicitSourceTerm(rpc*hco3mpspc/vwpc, var=hco3mcisf)
- ImplicitSourceTerm(hco3mpspc/vwpc, var=hco3mcpc) + ImplicitSourceTerm(cfpc*kf1, var=co2cpc)
- ImplicitSourceTerm(cfpc*kb1*hco3mcpc, var=hpcpc))

#Kopplung der Loesungsschritte der Differentialgleichungen
eqn = o2pl & o2isf & o2pc & co2pl & co2isf & co2pc & hppl & hpisf & hppc & hco3mpl & hco3misf & hco3mpc

#Durchlaufen der Zeitskala
while t<tmax:
    t = t+dt
    #Uebernahme der aktuellen Werte
    o2cpl.updateOld()
    o2cisf.updateOld()
    o2cpc.updateOld()
    co2cpl.updateOld()
    co2cisf.updateOld()
    co2cpc.updateOld()
    hpcpl.updateOld()
    hpcisf.updateOld()
    hpcpc.updateOld()
    hco3mcpl.updateOld()
    hco3mcisf.updateOld()
    hco3mcpc.updateOld()

    #Loesungsschritt fuer Dgl-System
    eqn.solve(dt=dt)

    #Aufzeichnen der neuen axialen Verteilung
    viO2.plot()
    viCO2.plot()
    viHp.plot()
    viHCO3m.plot()
```

A.3 MatLab-Programme

A.3.1 Sauerstoffmodell mit Differenzenquotient

```
function [O2Cplall, einflussparam, timeline] = o2tank3(modellparam, einflussparam, kO2, zeitparam)
%modellparam: n Spaltenvektoren: Volumen (1-4), Membranen (5-6), Enzym
%(7-8), L"oslichkeit (9), jeder Spaltenvektor f"ur einen Parametersatz
%einflussparam: Matrix mit Konzentrationen von O2 und Fluss als Spaltenvektor
%kO2: Sauerstoff"uberf"uhrungsvektor
%zeitparam: enth"alt tmax, dt, samplerate und vorlauf als Spaltenvektor

%Zeitparam
    if(nargin<4)
        zeitparam = [120;1e-3;1;240];
    end
    t = -zeitparam(4,1);
```

```
    tmax = zeitparam(1,1);
    dt = zeitparam(2,1);
    samplerate = zeitparam(3,1);

%"Uberf"uhrungsfaktoren
    if(nargin<3)
        kO2 = ones(1,tmax*samplerate+1);
    end

%Modellparameter
    if(nargin==0)
        modellparam = ones(1,1)*[0.07;0.94;0.2*0.92;0.7*0.8; 200; 2000; 2.5e-6;7e-7; 1.46e-6; 0.8*4e-4; 2.4]';
    end
    [n, n2] = size(modellparam);
    index = n2-n2+1;
    Vpl = modellparam(:,1)/1000;
    Wpl = modellparam(:,2);
    VWpl = Vpl.*Wpl;
    VWisf = modellparam(:,3)/1000;
    VWpc = modellparam(:,4)/1000;

    O2PScap = modellparam(:,5)/(60*1000);
    O2PSpc = modellparam(:,6)/(60*1000);

    Vmax = modellparam(:,7)/60;
    Km = modellparam(:,8);

    alphaO2 = modellparam(:,9);

    CMbpc = modellparam(:,10);
    P50Mb = modellparam(:,11)/1000;


%Einflussparameter
    if(nargin<2)
        einflussparam = o2einfluss(samplerate, tmax);
        %einflussparam = [6*ones(1,tmax*samplerate+1);634*ones(1,tmax*samplerate+1)];
    end
    F = einflussparam(1,:)/(60*1000);
    O2Cin = einflussparam(2,:)'*alphaO2'; %Spaltenvektor*Zeilenvektor -> Matrix

%Konzentrationen mit Startwert
    O2Cpl = O2Cin(index,:)';
    O2Cisf = O2Cin(index,:)';
    O2Cpc = O2Cin(index,:)';

%Speichervariablen f"ur R"uckgabe

    O2Cplall = zeros(n,tmax*samplerate+1);
    timeline = zeros(1,tmax*samplerate+1);

%Berechnung der Differentialgleichungen
    while(t<tmax)
        if(t>=(index-1)/samplerate)
            timeline(index) = t;
            O2Cplall(:,index) = O2Cpl./alphaO2;
            index = index+1;
        end
        t = t+dt;

        %Differentialgleichungen als Differenzenquotient
        a = -Vmax./(F(1,index)*Wpl);
        term = CMbpc.*alphaO2.*P50Mb./((O2Cpc+alphaO2.*P50Mb).^2-a.^2/4);
        O2Cpln = (kO2(1,index)*F(1,index)./Vpl.*(O2Cin(index,:)'-O2Cpl)-O2PScap./VWpl.*(O2Cpl-O2Cisf))*dt+O2Cpl;
        O2Cisfn = (O2PScap./VWisf.*(O2Cpl-O2Cisf)-O2PSpc./VWisf.*(O2Cisf-O2Cpc))*dt+O2Cisf;
        O2Cpcn = (O2PSpc./VWpc.*(O2Cisf-O2Cpc)-Vmax./VWpc.*O2Cpc./(Km+O2Cpc))*dt./(1+term)+O2Cpc;
        Gullo = Vmax./VWpc.*O2Cpc./(Km+O2Cpc);
        %"Ubertragung der Werte f"ur n"achsten Zeitschritt
        O2Cpl = O2Cpln;
        O2Cisf = O2Cisfn;
        O2Cpc = O2Cpcn;
    end
    timeline(index) = t;
    O2Cplall(:,index) = O2Cpl./alphaO2;
end
```

A.3.2 Metropolisalgorithmus auf Sauerstoffmodell gegen modellierte Daten

```
function [accept, avg, std2, vark, variat, acceptance,vpl,wpl,vwisf,vwpc,o2pscap,o2pspc,vmax,km,alphaO2,
cmb,p50mb] = metropolisO23(n, samplerate)
    tic;
    if(nargin==0)
        n = 100;
```

```
end
if(nargin <2)
   samplerate = 1;
end

tmax = 120;
modellparam = [0.07;0.94;0.2*0.92;0.7*0.8; 200; 2000; 2.5e-6;7e-7; 1.46e-6; 0.8*4e-4; 2.4];
einfluss = o2einfluss(samplerate, tmax);
zeitparam = [tmax;1e-3;samplerate;120];
err = 0.1;
%Funktion, die k-Werte einliest
if(samplerate==1)
    k = getkWerteO21();
elseif(samplerate==0.1)
    k=getkWerteO210();
else
    k = ones(1,tmax*samplerate+1);
end

%Funktion, die data vorgibt/generiert
[fulldata,gull] = o2tank3(modellparam', einfluss, k, zeitparam);
noisydata = noise2(fulldata,err,n);
noisygull = noise3(gull,n);
pdata = bayesmassO3(ones(n,1)*fulldata,noisydata, ones(n,1)*gull, noisygull, err);
modellparam = ones(n,1)*modellparam';
modellparam0 = modellparam;
pparam = paramwkO22(modellparam);

index = 0;
[versuche,params] = size(modellparam);
variat = modellparam(1,:)/10;
%warmup-zyklen
while(index<20)
    index = index+1;
    iter = 0;
    while(iter<params)
        iter = iter+1;
        modellparamn = modellparam;
        modellparamn(:,iter) = modellparamn(:,iter) + unifrnd(-variat(1,iter),variat(1,iter),versuche,1);
        modellparamn = abs(modellparamn);
        [modellresult, gullresult] = o2tank3(modellparamn, einfluss, k, zeitparam);
        %Sicherstellung der Modellstabilit"at
        %Ausgabe der Parameter bei Instabilit"at
        if(sum(isnan(modellresult))>0)
            iterator = 0;
            while(iterator<n)
                iterator = iterator+1;
                if(sum(isnan(modellresult(iterator,:)))>0)
                    bla2 = modellparamn(iterator,:)./modellparam0(iterator,:)
                end
            end
        end
        p1 = bayesmassO3(modellresult, noisydata, gullresult, noisygull, err);
        p2 = paramwkO22(modellparamn);
        r = exp(p2+p1-pparam-pdata);
        U = unifrnd(0,1,n,1);
        v = (r>=U);
        acceptance = sum(v);
        variat(1,iter) = variat(1,iter)*exp(acceptance/(n*0.35)-1.0);
        modellparam(:,iter) = (1-v).*modellparam(:,iter)+v.*modellparamn(:,iter);
        pparam = (1-v).*pparam+v.*p2;
        pdata = (1-v).*pdata+v.*p1;
    end
end

%regul"are Zyklen, deren Ergebnis gespeichert wird
steps = 500;
vpl = zeros(versuche,steps);
wpl = zeros(versuche,steps);
vwisf = zeros(versuche,steps);
vwpc = zeros(versuche,steps);
o2pscap = zeros(versuche,steps);
o2pspc = zeros(versuche,steps);
vmax = zeros(versuche,steps);
km = zeros(versuche,steps);
alphaO2 = zeros(versuche,steps);
cmb = zeros(versuche,steps);
p50mb = zeros(versuche,steps);
acceptance = zeros(params,steps);
index = 0;
while(index<steps)
    index = index+1;

    iter = 0;
    while(iter<params)
        iter = iter+1;
        modellparamn = modellparam;
        modellparamn(:,iter) = modellparamn(:,iter) + unifrnd(-variat(1,iter),variat(1,iter),versuche,1);
```

```
            modellparamn = abs(modellparamn);
            [modellresult, gulldata] = o2tank3(modellparamn, einfluss, k, zeitparam);

            %Sicherstellung der Modellstabilit"at
            %Ausgabe der Parameter bei Instabilit"at
            if(sum(isnan(modellresult))>0)
                iterator = 0;
                while(iterator<n)
                    iterator = iterator+1;
                    if(sum(isnan(modellresult(iterator,:)))>0)
                        bla2 = modellparamn(iterator,:)./modellparam0(iterator,:)
                    end
                end
            end

            p1 = bayesmassO3(modellresult, noisydata, gulldata, noisygull, err);
            p2 = paramwkO22(modellparamn);
            r = exp(p2+p1-pparam-pdata);
            U = unifrnd(0,1,n,1);
            v = (r>=U);
            acceptance(iter,index) = sum(v);
            %variat(1,iter) = variat(1,iter)*exp(acceptance(iter,index)/(n*0.35)-1.0);
            modellparam(:,iter) = (1-v).*modellparam(:,iter)+v.*modellparamn(:,iter);
            pparam = (1-v).*pparam+v.*p2;
            pdata = (1-v).*pdata+v.*p1;
        end

        vpl(:,index) = modellparam(:,1);
        wpl(:,index) = modellparam(:,2);
        vwisf(:,index) = modellparam(:,3);
        vwpc(:,index) = modellparam(:,4);
        o2pscap(:,index) = modellparam(:,5);
        o2pspc(:,index) = modellparam(:,6);
        vmax(:,index) = modellparam(:,7);
        km(:,index) = modellparam(:,8);
        alphaO2(:,index) = modellparam(:,9);
        cmb(:,index) = modellparam(:,10);
        p50mb(:,index) = modellparam(:,11);
    end

    accept = sum(acceptance,2)/steps;
    avg = zeros(params,1);
    std2 = ones(params,1);

    avg(1,1) = mean(vpl(:));
    std2(1,1) = std(vpl(:));

    avg(2,1) = mean(wpl(:));
    std2(2,1) = std(wpl(:));

    avg(3,1) = mean(vwisf(:));
    std2(3,1) = std(vwisf(:));

    avg(4,1) = mean(vwpc(:));
    std2(4,1) = std(vwpc(:));

    avg(5,1) = mean(o2pscap(:));
    std2(5,1) = std(o2pscap(:));

    avg(6,1) = mean(o2pspc(:));
    std2(6,1) = std(o2pspc(:));

    avg(7,1) = mean(vmax(:));
    std2(7,1) = std(vmax(:));

    avg(8,1) = mean(km(:));
    std2(8,1) = std(km(:));

    avg(9,1) = mean(alphaO2(:));
    std2(9,1) = std(alphaO2(:));

    avg(10,1) = mean(cmb(:));
    std2(10,1) = std(cmb(:));

    avg(11,1) = mean(p50mb(:));
    std2(11,1) = std(p50mb(:));

    vark = std2./avg;

    toc
end
```

A.3.3 Gesamtmodell mit Differenzenquotient

```
function [O2Cplall, CO2Cplall, HpCplall, HCO3mCplall, timeline] = tank9(modellparam, einflussparam, kfactorparam, zeitparam)
```

```
%modellparam: Spaltenvektor: Volumen (1-4), Membranen (5-16), Enzym
%(17-19), L"oslichkeit (20-21), Puffer (22-30)
%einflussparam: Matrix mit Einflusskonzentrationen f"ur alle Substrate als
%Spaltenvektor + Flussvektor
%kfactorparam: Matrix mit "Uberf"uhrungsfaktoren von Zylinder zu Tank-Modell
%zeitparam: enth"alt tmax, dt, samplerate und vorlauf als Spaltenvektor
    index = 1;
%Zeitparam
    if(nargin<4)
        zeitparam = [1200;1e-3;1;240];
    end
    t = -zeitparam(4,1);
    tmax = zeitparam(1,1);
    dt = zeitparam(2,1);
    samplerate = zeitparam(3,1);

%"Uberf"uhrungsfaktoren
    if(nargin<3)
        kfactorparam = ones(6,tmax*samplerate+1);
    end
    kO2 = kfactorparam(1,:);
    kCO2 = kfactorparam(2,:);
    kHCO3m = kfactorparam(3,:);
    kHp = kfactorparam(4,:);
    kH2PO4m = kfactorparam(5,:);
    kHPO42m = kfactorparam(6,:);

%Modellparameter
    if(nargin==0)
        modellparam = ones(100,1)*[0.07;0.94;0.2*0.92;0.7*0.8; 200; 2000; 200; 2000; 20;200;20;200;20;20;0.63;0.79;2.5e-6;7e-7;
        0.8; 100*0.12;5000*0.12;10000*0.12;0.12;6;24;45;0.8*4e-4; 2.4; 60]';
    end
    [n,n2] = size(modellparam);

    Vpl = modellparam(:,1)/1000;
    Wpl = modellparam(:,2);
    VWpl = Vpl.*Wpl;
    VWisf = modellparam(:,3)/1000;
    VWpc = modellparam(:,4)/1000;

    O2PScap = modellparam(:,5)/(60*1000);
    O2PSpc = modellparam(:,6)/(60*1000);
    CO2PScap = modellparam(:,7)/(60*1000);
    CO2PSpc = modellparam(:,8)/(60*1000);
    HCO3mPScap = modellparam(:,9)/(60*1000);
    HCO3mPSpc = modellparam(:,10)/(60*1000);
    HpPScap = modellparam(:,11)/(60*1000);
    HpPSpc = modellparam(:,12)/(60*1000);
    H2PO4mPScap = modellparam(:,13)/(60*1000);
    HPO42mPScap = modellparam(:,14)/(60*1000);

    Rcap = modellparam(:,15);
    Rpc = modellparam(:,16);

    Vmax = modellparam(:,17)/60;
    Km = modellparam(:,18);
    RQ = modellparam(:,19);

    alphaO2 = 1.46e-6;
    alphaCO2 = 3.27e-5;

    kf1 = 0.12;
    K1 = 4.47e-7;
    kb1 = kf1/K1;

    CFpl = modellparam(:,20)/kf1;
    CFisf = modellparam(:,21)/kf1;
    CFpc = modellparam(:,22)/kf1;

    kf2 = modellparam(:,23);
    K2 = 6.2e-8;
    kb2 = kf2/K2;

    BCpl = modellparam(:,24)/1000;
    BCisf = modellparam(:,25)/1000;
    BCpc = modellparam(:,26)/1000;

    CMbpc = modellparam(:,27);
    p50Mb = modellparam(:,28)/1000;

    H2PO4mCpc0 = modellparam(:,29)/1000;

%Einflussparameter
    if(nargin<2)
        einflussparam = [6*ones(1,tmax*samplerate+1);634*ones(1,tmax*samplerate+1);40*ones(1,tmax*samplerate+1);
        7.4*ones(1,tmax*samplerate+1); 1*ones(1,tmax*samplerate+1)];
    end
    F = einflussparam(1,:)/(60*1000);
```

```
    O2Cin = alphaO2*einflussparam(2,:);
    CO2Cin = alphaCO2*einflussparam(3,:);
    HpCin = 10.^(-einflussparam(4,:));
    HCO3mCin = CO2Cin(1,index)/HpCin(1,index)*K1;
    H2PO4mCin = einflussparam(5,:);
    HPO42mCin = H2PO4mCin(1,index)./HpCin(1,index)*K2;

%Konzentrationen mit Startwert
    O2Cpl = O2Cin(index);
    O2Cisf = O2Cin(index);
    O2Cpc = O2Cin(index);
    CO2Cpl = CO2Cin(index);
    CO2Cisf = CO2Cin(index);
    CO2Cpc = CO2Cin(index);
    HCO3mCpl = HCO3mCin(index);
    HCO3mCisf = HCO3mCin(index)./Rcap;
    HCO3mCpc = HCO3mCin(index)./(Rcap.*Rpc);
    HpCpl = HpCin(index);
    HpCisf = HpCin(index).*Rcap;
    HpCpc = HpCin(index).*Rcap.*Rpc;
    H2PO4mCpl = H2PO4mCin(index);
    H2PO4mCisf = H2PO4mCin(index)./Rcap;
    H2PO4mCpc = H2PO4mCpc0;
    HPO42mCpl = HPO42mCin(index);
    HPO42mCisf = HPO42mCin(index)./(Rcap.^2);
    HPO42mCpc = H2PO4mCpc0./HpCpc*K2;

%Speichervariablen f"ur R"uckgabe

    O2Cplall = zeros(n,tmax*samplerate+1);
    CO2Cplall = zeros(n,tmax*samplerate+1);
    HpCplall = zeros(n,tmax*samplerate+1);
    HCO3mCplall = zeros(n,tmax*samplerate+1);
    timeline = zeros(1,tmax*samplerate+1);

    prods = alphaO2*p50Mb;

%Berechnung der Differentialgleichungen
    while(t<tmax)
        if(t>=(index-1)/samplerate)
            timeline(index) = t;
            O2Cplall(:,index) = O2Cpl;
            CO2Cplall(:,index) = CO2Cpl;
            HCO3mCplall(:,index) = HCO3mCpl;
            HpCplall(:,index) = HpCpl;
            index = index+1;
        end
        t = t+dt;

        %Differentialgleichungen als Differenzenquotient
        a = - Vmax./(F(1,index).*Wpl);
        term = CMbpc.*prods./((O2Cpc+prods).^2-a.^2/4);

        O2Cpln = (kO2(1,index)*F(1,index)./Vpl.*(O2Cin(1,index)-O2Cpl)-O2PScap./VWpl.*(O2Cpl-O2Cisf))*dt+O2Cpl;
        O2Cisfn = (O2PScap./VWisf.*(O2Cpl-O2Cisf)-O2PSpc./VWisf.*(O2Cisf-O2Cpc))*dt+O2Cisf;
        O2Cpcn = (O2PSpc./VWpc.*(O2Cisf-O2Cpc)-Vmax./VWpc.*O2Cpc./(Km+O2Cpc))*dt./(1+term)+O2Cpc;

        CO2Cpln = (kCO2(1,index)*F(1,index)./Vpl.*(CO2Cin(1,index)-CO2Cpl-CO2PScap./VWpl.*(CO2Cpl-CO2Cisf)
        -CFpl.*(kf1.*CO2Cpl-kb1.*HCO3mCpl.*HpCpl))*dt+CO2Cpl;
        CO2Cisfn = (CO2PScap./VWisf.*(CO2Cpl-CO2Cisf)-CO2PSpc./VWisf.*(CO2Cisf-CO2Cpc)
        -CFisf.*(kf1.*CO2Cisf-kb1.*HCO3mCisf.*HpCisf))*dt+CO2Cisf;
        CO2Cpcn = (CO2PSpc./VWpc.*(CO2Cisf-CO2Cpc)+RQ.*Vmax./VWpc.*O2Cpc./(Km+O2Cpc)
        -CFpc.*(kf1.*CO2Cpc-kb1.*HCO3mCpc.*HpCpc))*dt+CO2Cpc;

        HCO3mCpln = (kHCO3m(1,index)*F(1,index)./Vpl.*(HCO3mCin-HCO3mCpl)-HCO3mPScap./VWpl.*(Rcap.*HCO3mCpl-HCO3mCisf)
        +CFpl.*(kf1.*CO2Cpl-kb1.*HCO3mCpl.*HpCpl))*dt+HCO3mCpl;
        HCO3mCisfn = (HCO3mPScap./VWisf.*(Rcap.*HCO3mCpl-HCO3mCisf)
        -HCO3mPSpc./VWisf.*(Rpc.*HCO3mCisf-HCO3mCpc)
        +CFisf.*(kf1.*CO2Cisf-kb1.*HCO3mCisf.*HpCisf))*dt+HCO3mCisf;
        HCO3mCpcn = (HCO3mPSpc./VWpc.*(Rpc.*HCO3mCisf-HCO3mCpc)+CFpc.*(kf1.*CO2Cpc-kb1.*HCO3mCpc.*HpCpc))*dt+HCO3mCpc;

        HpCpln = (kHp(1,index)*F(1,index)./Vpl.*(HpCin(1,index)-HpCpl)-HpPScap./VWpl.*(HpCpl-Rcap.*HpCisf)
        +2.303./BCpl.*HpCpl.*(CFpl.*(kf1.*CO2Cpl-kb1.*HCO3mCpl.*HpCpl)
        +(kf2.*H2PO4mCpl-kb2.*HPO42mCpl.*HpCpl)))*dt+HpCpl;
        HpCisfn = (HpPScap./VWisf.*(HpCpl-Rcap.*HpCisf)-HpPSpc./VWisf.*(HpCisf-Rpc.*HpCpc)
        +2.303./BCisf.*HpCisf.*(CFisf.*(kf1.*CO2Cisf-kb1.*HCO3mCisf.*HpCisf)
        +(kf2.*H2PO4mCisf-kb2.*HPO42mCisf.*HpCisf)))*dt+HpCisf;
        HpCpcn = (HpPSpc./VWpc.*(HpCisf-Rpc.*HpCpc)
        +2.303./BCpc.*HpCpc.*(CFpc.*(kf1.*CO2Cpc-kb1.*HCO3mCpc.*HpCpc)
        +(kf2.*H2PO4mCpc-kb2.*HPO42mCpc.*HpCpc)))*dt+HpCpc;

        H2PO4mCpln = (kH2PO4m(1,index)*F(1,index)./Vpl.*(H2PO4mCin(1,index)-H2PO4mCpl)
        -H2PO4mPScap./VWpl.*(Rcap.*H2PO4mCpl-H2PO4mCisf)
        -(kf2.*H2PO4mCpl-kb2.*HPO42mCpl.*HpCpl))*dt+H2PO4mCpl;
        H2PO4mCisfn = (H2PO4mPScap./VWisf.*(Rcap.*H2PO4mCpl-H2PO4mCisf)
        -(kf2.*H2PO4mCisf-kb2.*HPO42mCisf.*HpCisf))*dt+H2PO4mCisf;
        H2PO4mCpcn = -(kf2.*H2PO4mCpc-kb2.*HPO42mCpc.*HpCpc)*dt+H2PO4mCpc;
```

```
        HPO42mCpln = (kHPO42m(1,index)*F(1,index)./Vpl.*(HPO42mCin-HPO42mCpl)
        -HPO42mPScap./VWpl.*(Rcap.^2.*HPO42mCpl-HPO42mCisf)
        +(kf2.*H2PO4mCpl-kb2.*HPO42mCpl.*HpCpl))*dt+HPO42mCpl;
        HPO42mCisfn = (HPO42mPScap./VWisf.*(Rcap.^2.*HPO42mCpl-HPO42mCisf)
        +(kf2.*H2PO4mCisf-kb2.*HPO42mCisf.*HpCisf))*dt+HPO42mCisf;
        HPO42mCpcn = (kf2.*H2PO4mCpc-kb2.*HPO42mCpc.*HpCpc)*dt+HPO42mCpc;

        %"Ubertragung der Werte f"ur n"achsten Zeitschritt
        O2Cpl = O2Cpln;
        O2Cisf = O2Cisfn;
        O2Cpc = O2Cpcn;

        CO2Cpl = CO2Cpln;
        CO2Cisf = CO2Cisfn;
        CO2Cpc = CO2Cpcn;

        HCO3mCpl = HCO3mCpln;
        HCO3mCisf = HCO3mCisfn;
        HCO3mCpc = HCO3mCpcn;

        HpCpl = HpCpln;
        HpCisf = HpCisfn;
        HpCpc = HpCpcn;

        H2PO4mCpl = H2PO4mCpln;
        H2PO4mCisf = H2PO4mCisfn;
        H2PO4mCpc = H2PO4mCpcn;

        HPO42mCpl = HPO42mCpln;
        HPO42mCisf = HPO42mCisfn;
        HPO42mCpc = HPO42mCpcn;
    end
    timeline(index) = t;
    O2Cplall(:,index) = O2Cpl;
    CO2Cplall(:,index) = CO2Cpl;
    HCO3mCplall(:,index) = HCO3mCpl;
    HpCplall(:,index) = HpCpl;

    O2Cplall = O2Cplall/alphaO2;
    CO2Cplall = CO2Cplall/alphaCO2;
    HpCplall = -log10(HpCplall);
end
```

A.3.4 Metropolisalgorithmus im Vollmodell für einzelnen Parameter

```
function [param, variat, acceptance] = metropolis3(n, iter, samplerate,steps)
    tic;
    if(nargin==0)
        n = 100;
    end
    if(nargin <2)
        iter=1;
    end
    if(nargin <3)
       samplerate = 1;
    end
    if(nargin<4)
        steps = 3000;
    end

    tmax = 1200;
    modellparam = [0.07;0.94;0.2*0.92;0.7*0.8; 200; 2000; 200; 2000; 20;200;20;200;20;20;0.63;0.79;2.5e-6;7e-7;
    0.8; 100*0.12;5000*0.12;10000*0.12;0.12;6;24;45;0.8*4e-4; 2.4; 60];
    einfluss = experimenteinfluss(samplerate);
    zeitparam = [tmax;1e-3;samplerate;300];
    err = [0.1;0.1;0.005;0.0001];
    %Funktion, die k-Werte einliest
    if(samplerate==1)
        k = getkWerte1();
    elseif(samplerate==0.1)
        k=getkWerte10();
    else
        k = ones(6,tmax*samplerate+1);
    end
    %Funktion, die data vorgibt/generiert
    [o2,co2,hp,hco3m] = tank9(modellparam', einfluss, k, zeitparam);
    [o2e, co2e, hpe, hco3me] = fullnoise2(o2,co2,hp,hco3m,err,n);
    pdata = wkdata(ones(n,1)*o2,ones(n,1)*co2,ones(n,1)*hp,ones(n,1)*hco3m, o2e, co2e, hpe, hco3me, err);
    modellparam = ones(n,1)*modellparam';
    modellparam0 = modellparam;
    pparam = wkparam2(modellparam);

    index = 0;
    [versuche,params] = size(modellparam);
    variat = modellparam(1,:)/10;
```

```
    %warmup-zyklen
    while(index<10)
        index = index+1;
        modellparamn = modellparam;
        modellparamn(:,iter) = variatmodellparam(modellparam(:,iter), variat(1,iter), modellparam0(1,iter));
        [o2, co2, hp, hco3m] = tank9(modellparamn, einfluss, k, zeitparam);
        %Sicherstellung der Modellstabilit"at, sonst Ausgabe, bei welcher
        %Parameterwahl der Fehler eintritt
        if(sum(isnan(co2))>0)
            zeitparam2 = zeitparam;
            zeitparam2(2) = 1e-4;
            [o2,co2,hp,hco3m] = tank9(modellparamn, einfluss, k, zeitparam2);
            testung = (isnan(co2)+isinf(co2));
            if(sum(testung)>0)
                iterator = 0;
                while(iterator<n)
                    iterator = iterator+1;
                    if(sum(testung(iterator,:))>0)
                        bla2 = modellparamn(iterator,:)./modellparam0(iterator,:)
                        o2(iterator,:) = zeros(1, tmax*samplerate+1);
                        co2(iterator,:) = zeros(1, tmax*samplerate+1);
                        hp(iterator,:) = zeros(1, tmax*samplerate+1);
                        hco3m(iterator,:) = zeros(1, tmax*samplerate+1);
                    end
                end
            end
        end
        p1 = wkdata(o2, co2, hp, hco3m, o2e, co2e, hpe, hco3me, err);
        p2 = wkparam2(modellparamn);
        r1 = min(p2+p1-pparam-pdata,1);
        r = exp(r1);
        U = unifrnd(0,1,n,1);
        v = (r>=U);
        accept = sum(v);
        variat(1,iter) = variat(1,iter)*exp(accept/(n*0.35)-1.0);
        modellparam(:,iter) = (1-v).*modellparam(:,iter)+v.*modellparamn(:,iter);
        pparam = (1-v).*pparam+v.*p2;
        pdata = (1-v).*pdata+v.*p1;
    end
%regul"are Zyklen, deren Ergebnis gespeichert wird
    param = zeros(versuche,steps);
    acceptance = zeros(params,steps);
    index = 0;
    while(index<steps)
        index = index+1;
        modellparamn = modellparam;
        modellparamn(:,iter) = variatmodellparam(modellparam(:,iter), variat(1,iter), modellparam0(1,iter));
        [o2, co2, hp, hco3m] = tank9(modellparamn, einfluss, k, zeitparam);
        %Sicherstellung der Modellstabilit"at
        %Ausgabe, bei welcher Parameterwahl Fehler auftraten
        if(sum(isnan(co2))>0)
            zeitparam2 = zeitparam;
            zeitparam2(2) = 1e-4;
            [o2,co2,hp,hco3m] = tank9(modellparamn, einfluss, k, zeitparam2);
            testung = (isnan(co2)+isinf(co2));
            if(sum(testung)>0)
                iterator = 0;
                while(iterator<n)
                    iterator = iterator+1;
                    if(sum(testung(iterator,:))>0)
                        bla2 = modellparamn(iterator,:)./modellparam0(iterator,:)
                        o2(iterator,:) = zeros(1, tmax*samplerate+1);
                        co2(iterator,:) = zeros(1, tmax*samplerate+1);
                        hp(iterator,:) = zeros(1, tmax*samplerate+1);
                        hco3m(iterator,:) = zeros(1, tmax*samplerate+1);
                    end
                end
            end
        end
        p1 = wkdata(o2, co2, hp, hco3m, o2e, co2e, hpe, hco3me, err);
        p2 = wkparam2(modellparamn);
        r1 = min(p2+p1-pparam-pdata,1);
        r = exp(r1);
        U = unifrnd(0,1,n,1);
        v = (r>=U);
        acceptance(iter, index) = sum(v);
        modellparam(:,iter) = (1-v).*modellparam(:,iter)+v.*modellparamn(:,iter);
        pparam = (1-v).*pparam+v.*p2;
        pdata = (1-v).*pdata+v.*p1;
        param(:,index) = modellparam(:,iter);
    end

    toc
end
```

A.3.5 Sauerstoffsystem für ODE-Solver

```
function dy = expGul(t,y,param, tf, Fin, kin)
%Funktion, die einen einzelnen Zeitschritt des Solvers berechnet und die 3
%Konzentrationen zur"uckgibt
dy = zeros(3,1);    % a column vector
Vpl = param(1)/1000;
Wpl = param(2);
VWpl = Vpl*Wpl;
VWisf = param(3)/1000;
VWpc = param(4)/1000;
O2PScap = param(5)/(60*1000);
O2PSpc = param(6)/(60*1000);
Vmax = param(7)/60;
Km = param(8);
alphaO2 = param(9);
CMbpc = param(10);
P50Mb = param(11)/1000;

O2Cin = 634*alphaO2;
F = interp1(tf, Fin, t)/(60*1000);
k = interp1(tf, kin, t);
a = -Vmax/(F*Wpl);
term = CMbpc*alphaO2*P50Mb/((y(3)+alphaO2*P50Mb)^2-a^2/4);

dy(1) = k*F/Vpl*(O2Cin-y(1))-O2PScap/VWpl*(y(1)-y(2));
dy(2) = O2PScap/VWisf*(y(1)-y(2))-O2PSpc/VWisf*(y(2)-y(3));
dy(3) = (O2PSpc/VWpc*(y(2)-y(3))-Vmax/VWpc*y(3)/(Km+y(3)))/(1+term);
```

A.3.6 Sauerstoffsystem ODE-Solver

```
function [T,Y] = calculateGul(modellparam, t, Fin, kin, options)
%Funktion, die mit ode-Solver Konzentrationen von Sauerstoff im zeitlichen Verlauf berechnet
    if(nargin<5)
        options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
    end
    alphaO2 = modellparam(9);
    [T,Y] = ode15s(@(T,Y) expGul(T, Y, modellparam, t, Fin, kin),t,alphaO2*[634 634 634],options);
end
```

A.3.7 Metropolisalgorithmus gegen experimentellen Stoffumsatz

```
function [finalparams, accepts, r, accept] = metroGul(t, Fin, kin, Gul, err, param)
    tic;
    if(nargin<6)
        param = [0.07;0.94;0.2*0.92;0.7*0.8; 200; 2000; 2.5e-6;7e-7; 1.46e-6; 0.8*4e-4; 2.4];
    end
    if(nargin<5)
        err = 0.1;
    end
    options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
    [T,Y] = calculateGul(param, t, Fin, kin, options);
    b = -param(7)./(Fin*param(2))*1000;
    term = param(10)*param(9)*param(11)./((Y(:,3)'+param(9)*param(11)).^2-b.^2/4);
    GulE = (param(7)*Y(:,3)'./(param(8)+Y(:,3)'))./(1+term)*1e6;
    wkparam = wkparamfluss(param);
    wkdata = dataprob(t, Gul, GulE, err);

    variance = param/100;
    zyklen = 0;
    accept = zeros(11,30);
    while(zyklen<30)
        zyklen = zyklen+1;
        iter = 4;
        while(iter<11)
            iter = iter+1;
            counter = 0;
            while(counter<100)
                counter = counter+1;
                paramn = param;
                paramn(iter) = abs(param(iter) + unifrnd(-variance(iter), variance(iter), 1, 1));
                [T,Y] = calculateGul(paramn, t, Fin, kin, options);
                %Berechnung von Gulosity gem"a"s der Formel f"ur
                %Myoglobinkorrektur
                b = -paramn(7)./(Fin*paramn(2))*1000;
                term = paramn(10)*paramn(9)*paramn(11)./((Y(:,3)'+paramn(9)*paramn(11)).^2-b.^2/4);
                GulE = (paramn(7)*Y(:,3)'./(paramn(8)+Y(:,3)'))./(1+term)*1e6;
                p1 = wkparamfluss(paramn);
                p2 = dataprob(t,Gul,GulE, err);
                r = exp(min(0,p1+p2-wkparam-wkdata));
                s = unifrnd(0,1,1,1);
```

```
                a = (r>s);
                accept(iter,zyklen) = accept(iter,zyklen)+a;
                param = param*(1-a)+a*paramn;
                wkparam = wkparam*(1-a)+a*p1;
                wkdata = wkdata*(1-a)+a*p2;
            end
            variance(iter) = variance(iter)*exp(accept(iter,zyklen)/35-1.0);
        end
    end
    steps = 1;
    finalparams = zeros(11,5000);
    accepts = zeros(11,1);
    finalparams(:,1) = param;
    r = zeros(11,5000);
    while(steps<5000)
        steps = steps+1;
        iter = 4;
        while(iter<11)
            iter = iter+1;
            paramn = param;
            paramn(iter) = abs(param(iter) + unifrnd(-variance(iter), variance(iter), 1, 1));
            [T,Y] = calculateGul(paramn, t, Fin, kin, options);
            %Berechnung von Gulosity gem"a"s der Formel f"ur
            %Myoglobinkorrektur
            b = -paramn(7)./(Fin*paramn(2))*1000;
            term = paramn(10)*paramn(9)*paramn(11)./((Y(:,3)'+paramn(9)*paramn(11)).^2-b.^2/4);
            GulE = (paramn(7)*Y(:,3)'./(paramn(8)+Y(:,3)'))./(1+term)*1e6;
            p1 = wkparamfluss(paramn);
            p2 = dataprob(t,Gul,GulE, err);
            r(iter,steps) = exp(min(0,p1+p2-wkparam-wkdata));
            s = unifrnd(0,1,1,1);
            a = (r(iter,steps)>s);
            accepts(iter) = accepts(iter)+a;
            param = param*(1-a)+a*paramn;
            wkparam = wkparam*(1-a)+a*p1;
            wkdata = wkdata*(1-a)+a*p2;
        end
        finalparams(:,steps) = param;
    end
    toc
end
```

Literaturverzeichnis

[1] Austin, W.H., E.Lacombe, P. W. Rand, and M. Chatterjee. Solubility of carbon dioxide in serum from 15 to 38°C. J.Appl.Physiol., 18:301-304,1963

[2] Bassingthwaighte, J.B Plasma indicator dispersion in arteries of the human leg. Circ. Res., 19:332-346,1966

[3] Bassingthwaighte, J. B., and C. A. Gorensky. Modeling in the analysis of solute and water exchange in the microvasculature. In: Handbook of Physiology. Sect 2, The Cardiovascular System. Vol IV, The Microcirculation, edited by E. M. Renkin and C.C. Michel. Bethesda, MD: Am. Physiol. Soc., 549-626,1984

[4] Bassingthwaighte, J.B., F.P.Chinard, C.Crone, C.A.Gorensky, N.A.Lassen, R.S.Reneman, and K.L.Zierler. Terminology for mass transport and exchange. Am. J. Physiol. Heart Circ. Physiol., 250:H539-H545, 1986

[5] Beard, D.A., and J.B. Bassingthwaighte. Advection and diffusion of substances in biological tissues with complex vascular networks. Ann. Biomed. Eng., 29:298-310, 2001

[6] Buck, J., L.R. Levin. Physiological Sensing of Carbon Dioxide/Bicarbonate/pH via Cyclic Nucleotide Signaling, Sensors, 11:2112-2128, 2011

[7] Dash, R.K. and J.B. Bassingthwaighte. Simultaneous Blood-Tissue Exchange of Oxygen, Carbon Dioxide, Bicarbonate, and Hydrogen Ion, Ann Biomed Eng 34(7), 1129-1148, 2006

[8] Deussen,A., and J.B.Bassingthwaithe. Modeling[^{15}O]oxygen tracer data for estimating oxygen consumption. Am. J. Physiol. Heart Circ. Physio., 270:H1115-H1130, 1996

[9] Dieterich, P., A. Deussen, Seminar zur Mathematischen Physiologie, Bayessche Datenanalyse, Dresden, 2013

[10] Dieterich, P., Modellierung von pCO2-Gewebeprofilen am Herzen, Vortrag beim 2. Xantener Gespräch zur Medizinisch-Biologischen Chemie 'Von molekularen Wechselwirkungen zu Lebensvorgängen', Xanten, 2004

[11] Dose, V. Bayesian inference in physics: case studies, Institute of Physics Publishing, 2003

[12] Engelhardt, W. and G.Breeves. Physiologie der Haustiere, Enke Verlag, Stuttgart, 3. Auflage, 284-288, 2010

[13] Guyer, J.E., D. Wheeler, J.A. Warren, Fipy: Partial Differential Equations with Python, Computing in Science & Engineering, 11:6-15, 2009

[14] Goldman,D., and A.S.Popel. A computational study of the effect of capillary network anastomoses and totuosity on oxygen transport. J.Theor.Biol., 206:181-194, 2000

[15] Gregory, P.C., Bayesian Logical Data Analysis for the Physical Sciences, Cambridge University Press, 2005

[16] Hedley-Whyte, J. and M.B. Laver. O_2 Solubility in blood and temperature correction factors for PO_2. J.Appl.Physiol.,19:901-906, 1964

[17] Heintz, A., T.Koch, and A.Deussen, Intact nitric oxide production is obligatory for the sustained flow response during hypercapnic acidosis in guinea pig heart, Cariovascular Research, 66:55-63, 2005

[18] Heintz, A., M. Damm, M. Brand, T. Koch, A. Deussen. Coronary flow regulation in mouse heart during hypercapnic acidosis: role of NO and its compensation during eNOS impairment, European Society of Cardiology, 2008

[19] Hill, E. P., G.G Power and L.D.Longo. A mathematical model of carbon dioxide transfer in the placenta and its interaction with oxygen. Am. J.Physiol.Cell Physiol.,224:283-299, 1973

[20] Hill, E.P., G.G.Power, and L.D.Longo. Mathematical simulaion of pulmonary O_2 and CO_2 exchange. Am. J.Physiol.Cell Physiol.,224:904-917, 1973

[21] Hill, E.P., G.G.Power, and L.D.Longo. Kinetics of O_2 and CO_2 exchange. In: Bioengineering Aspects of the Lung, edited by J.B. West. New York: Marcel Dekker, 459-514, 1977

[22] Huang, N.S., and J.D. Hellums. A theoretical model for gas transport and acid/base regulation by blood flowing in microvessels. Microvasc. Res., 48:364-388, 1994

[23] King, R., A. Deussen, G. Raymond, J.B. Bassingthwaighte. A vascular transport operator. Am. J.Physiol. 265 H2196-H2208, 1993

[24] Klipp, E., R. Herwig, A. Kowald, C. Wierling, H. Lehrach. Systems Biology in Practice, Wiley-VCH Verlag, Weinheim, 2005

[25] Lange, T., Weiterentwicklung und Simulation eines mathematischen Modells für

den O_2 und CO_2-Transport am isoliert perfundierten Herzen, Praktikumsarbeit, 2013

[26] Li, Z., T. Yipintsoi, and J.B.Bassingthwaighte. Nonlinear model for capillary-tissue oxygen transport and metabolism. Ann. Biomed. Eng., 25:604-619, 1997

[27] Thieme Chemistry,RÖMPP Online - Version 3.5, Georg Thieme Verlag KG, Stuttgart, 2009

[28] Safford,R.E., and J.B. Bassingthwaighte. Calcium diffusion in transient and steady states in muscle. Biophys. J., 20:113-136, 1977

[29] Salathe, E.P.,R.Fayad, and SW. Schaffer. Mathematical analysis of carbon dioxide transfer by blood. Math. Biosci., 57:109-153, 1981

[30] Schmidt,F.,G.Thews, and F.Lang. Physiologie des Menschen, 28. Auflage, Springer Verlag, New York, 2000

[31] Schuster,R., Biomathematik - Mathematische Modelle in der Medizinischen Informatik und in den Computational Life Sciences, Vieweg-Teubner-Verlag, Wiesbaden, 1. Auflage, 95-118, 2009

[32] Schenkman, K.A.,D.R. Marble, D.H.Burns, and E.O. Feigl. Myoglobin oxygen dissociation by multiwavelength spectroscopy. J.Appl.Physiol.,82:86-92, 1997

[33] Schenkman,K.A,D.R.Marble, D.H.Burns, and E.O.Feigl. Optical spectroscopic method for in vivo measurement of cardiac myoglobin oxygen saturation. Appl. Spectrosc.,53:332-338, 1999

[34] Suenson, M.,D.R.Richmond, and J.B. Bassingthwaighte. Diffusion of sucrose, sodium, and water in ventricular myocardium. Am. J. Physiol., 227:1116-1123, 1974

[35] Tresguerres, M., J. Buck, L.R. Levin. Physiological carbon dioxide, bicarbonate and pH sensing, Eur J Physiol, 460:953, 2010

[36] Tresguerres, M., L.R. Levin, J. Buck. Intracellular cAMP signaling by soluble adenylyl cyclase, Kidney International, 79:1277-1288, 2011

[37] Winslow, R. M., M.Samaja, N. J. Winslow, L. Rossi Bernardi, and R.I. Shrager, Simulation of continuous blood O_2 equilibrium over physiological pH, DPG, and pCO_2 range. J. Appl.Physiol.:Respir. Environ.Exercise Physiol., 54:524-529, 1983

[38] www.mathworks.de/products/matlab, The MathWorks, Inc., abgerufen am 07.06.2013

[39] www.physiome.org, Bassingthwaighte, J.B., Physiome Project, abgerufen am 05.06.2013

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 28. August 2013